Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Editor i Disseny d'Interiors

Francesc Ros Olivé
Jordi Serra Gil

Enginyeria Tècnica en Informàtica de Gestió

Director: Jordi Surinyac

Vic, febrer de 2007

Índex

Universitat de Vic
Escola Politècnica Superior

Resum de Treball Final de Carrera
Enginyeria Tècnica d'Informàtica de Gestió

Títol: Editor i Disseny d'Interiors.
Paraules clau: Editor interiors, disseny interiors, mobles, OpenGL, GLUT.
Autors: Francesc Ros Olivé i Jordi Serra Gil
Direcció: Jordi Surinyac
Data: febrer de 2007

Resum

L'OpenGL és un motor 3D que s'utilitza com a lligam entre el software i el hardware gràfic. Actualment és una de les tecnologies més utilitzades en el disseny d'aplicacions 3D.
El treball està realitzat amb el programa Visual C++, que és el més recomanat per al desenvolupament d'aplicacions OpenGL.
L'objectiu principal d'aquest treball és aprendre a programar amb aquest tipus de tecnologia que no hem estudiat durant el període de la carrera.
Un altre objectiu del treball era trobar una funció útil i pràctica per a l'aplicació i ens vam decantar per a realitzar un editor d'habitacions per una botiga o empresa de mobles.
L'usuari pot de forma molt ràpida i senzilla dibuixar com és l'habitació que vol decorar de forma totalment personalitzada. El programa li generarà l'habitació en tres dimensions i amb els materials que s'han escollit (terra, parets, portes...).
Després pot editar-hi mobles personalitzats o pertanyents a la llibreria del programa.
El programa incorpora també una base de dades per a l'empresa que ens portarà la gestió de clients, habitacions, textures i mobles (permet ampliar la llibreria del programa).
Un cop acabada l'habitació el programa ens hi permet fer una visita de forma interactiva i generar-ne la factura entre altres funcions.
La conclusió principal després d'haver acabat aquest projecte, és que a part d'haver après OpenGL, hem aconseguit realitzar una aplicació molt pràctica de cares al disseny d'interiorisme.

Universitat de Vic
Escola Politècnica Superior

Summary of the Final Project

Title: Interiorism Editor and Designer.
Keywords: Interiorism editor, disseny interiors, mobles, OpenGL, GLUT.
Authors: Francesc Ros Olivé i Jordi Serra Gil
Direction: Jordi Surinyac
Date: February, 2007

Summary

OpenGl is a 3D motor used as a link between software and graphic hardware. Nowadays is one of the most used tecnologies in 3D application design.
The paper has been done using C++ program, the most recommended one in OpenGl applications development.

the main objective of this paper is to learn to program with a tecnology that we haven't studied during our learning period in our degree.

another objective was to find an useful and practical function for the application, so we decided to create a room editor thinking to be used in a forniture company or in a shop.
The user can draw in a very easy and fast way how he wants to decorate the room in a total personal way. The program generates the room in 3D with the chosen materials (floor, walls, doors,...).
Then he can edit personalized furniture pieces or can choose the ones in the program's library. The program also incorporates a data base fort he company. It is going to bring as to the costumer, rooms, textures and furnitures management. It allows as to enlarge the library and when the room is finished the program invites as to do an interactive tour and generate the bill among other facilities.

The main conclusion after having finished the project is that we have learned how to use OpenGL and we have also create a very practical aplication for the interior design.

Introducció

El projecte surt de la idea d'aprofitar la llibertat d'elecció del tema del TFC.
Els dos volíem aprofitar-ho per fer una cosa no apresa durant el període de la carrera i així assolir nous coneixements.
Vam estar demanant idees als professors per tal de trobar alguna cosa nova i que ens motivés.
Parlant amb en Jordi Surinyac ens va proposar entre altres coses, el tema de l'OpenGL i ens va semblar que era molt interessant.
Ens faltava saber què fèiem amb OpenGL, les possibilitats són molt àmplies, des de simuladors (vam estar estudiant la possibilitat de fer un simulador d'ala delta), a programes estadístics (controlar semàfors per tal de veure com corria el trànsit) passant per jocs i altres aplicacions.
Finalment, vam decidir que havia de ser una cosa més pràctica, realment útil i oberta a qualsevol usuari o persona.
Va sorgir la idea de l'editor de mobles o habitacions. Vam pensar que la utilitat era gran ja que pot resultar difícil canviar el mobiliari d'una habitació sense poder-ne tenir una imatge global de com pot quedar un cop comprats els mobles (problemes d'espais, estètics, etc.).
Hi ha la possibilitat de crear una habitació renderitzada amb programes d'entorn 3D (3DStudio Max, AutoCad, Rhino...), però són treballs molt elaborats, que porten hores i hores de feina i a més requereixen un gran nivell pràctic per part de la persona que els realitza.
La nostra aplicació per contra, la podria utilitzar qualsevol persona i encara que no ofereix el gran nivell de detall que pot oferir un render, es pot fer molt ràpid i se'n treu una imatge final prou realista com per a que te'n puguis fer una idea de com quedaria l'habitació. Oferint a més, una interacció amb l'usuari i un dinamisme que no ofereixen els renders.
Al moment d'exposar-li a en Jordi Surinyac, la idea li va semblar molt bé, va dir que era possible tirar-la endavant, i que el projecte era prou gran com per abarcar el treball de dues persones.

El projecte permet una total personalització per part de l'usuari. L'habitació s'introdueix amb un editor de dibuix creat a mesura per tal de realitzar aquesta funció, el programa genera l'espai 3D amb les dades obtingudes i després es poden afegir mobles, moure'ls, rotar-los, esborrar-los....Tant de mobles fets a mida per el propi usuari com mobles creats prèviament per programes d'entorn 3D (mitjançant el format .ASE).

A més el projecte permet una simulació en primera persona d'una visita per dins l'habitació per tal d'obtenir-ne una bona imatge de les proporcions dels mobles dins l'espai. També podem obtenir una visió més general a partir d'una sèrie de vistes en diferents perspectives que són també modificables al gust de l'usuari.
Per tal d'englobar tot el contingut el projecte inclou també una base de dades que porta la gestió tant de clients, com d'habitacions, preus, stock, etc.

Finalment, no volíem deixar el projecte tancat, i per tant, vam adaptar-lo per tal de que el propietari del projecte pogués afegir-hi nous mobles i nous materials o textures.

Hem estructurat la memòria en 3 parts perquè aquestes són totalment independents i per a fer-ho lo més entenedor possible per al lector.

Requeriments tècnics

L'OpenGL ens permet crear escenes en tres dimensions i interactives, utilitzant una sèrie de llibreries. L'OpenGL mereix una explicació més detallada que comentarem més endavant[1].
Per tal de manipular-les podem utilitzar gran varietat de llenguatges, tot i que es recomana especialment l'ús del Microsoft Visual C++.

El Visual C++ permet programar utilitzant l'MFC (Microsoft Fundation Classes), que és una jerarquia de classes. Aquestes classes faciliten la programació Windows, sense haver d'utilitzar directament l'API de Windows, ja que agrupen les llibreries de Windows en classes C++. MFC ens dóna un model de desenvolupament d'aplicacions anomenat model document/vista, que ens permet dissenyar aplicacions de forma que les dades de l'aplicació vagin separades dels elements que componen la interfície d'usuari. Això permet modificar de forma independent les dues parts del programa.

Per a la base de dades, hem utilitzat el Microsoft Acces per tal d'emmagatzemar-hi la informació i per lligar aquesta informació amb el Visual C++ hem fet servir l'ADO (ActiveX Data Objects), que és un mecanisme que serveix per donar ordres i obtenir resultats d'elles, permetent manipular la base de dades (altes, baixes, modificacions,...). Explicarem més detalladament aquest mecanisme més endavant[2].

Per a la creació dels mobles no modificables, hem utilitzat el format ASE. És un format de tipus text (ASCII) que conté la informació d'un objecte creat amb un programa d'entorn 3D. En el nostre cas hem utilitzat el programa 3DStudio Max, que permet la creació d'objectes en tres dimensions amb un gran nivell de detall (hi ha objectes que poden tenir més de 40.000 cares) i exportar-les en aquest format, donant gran quantitat d'informació.

A més també hem utilitzat per al retoc de textures l'Adobe Photoshop.

[1] II.2.- Introducció a OpenGL pàg. 46
[2] III.2.-ADO pàg.122

Els projectes en Visual C++ i el problema de la comunicació:

L'aplicació general del projecte, és un WorkSpace del Visual C++ (l'espai de treball), format per varis projectes. Podríem definir els diferents projectes com una sèrie de "paquets" o petites aplicacions, que al final ajuntats composen el projecte final.

Des d'un principi això representava una sèrie d'avantatges, podíem anar fent la feina pas a pas i per parts, treballant amb coses diferents i a més al ser dos persones, podíem treballar en projectes diferents i així no teníem problemes alhora d'ajuntar el projecte.

Això però comportava un problema, i era la comunicació entre projectes. Cada un dels projectes està tancat, i cada un té el seu propi arxiu.exe.

Els diferents projectes que formen el programa final són:

- El projecte Base de Dades: És tota la part de la base de dades pròpiament dita, és el projecte que s'executa com a principal i des d'on anirem executant la resta de projectes.
- El projecte Dibuixar: És l'editor de dibuix per tal de crear l'habitació i qui genera el fitxer que després utilitzarà l'OpenGL per a la creació de l'espai 3D.
- El projecte Vista_Planta: És l'editor d'habitacions, el més important i alhora més complicat del TFC. És on es treballa amb l'habitació i els mobles, i on es poden fer tot tipus d'accions sobre ells com afegir-ne, moure'n, rotar-ne, etc.
- El projecte Carregador ASE: És l'encarregat de transformar un fitxer ASE a un de TXT preparat per a ser interpretat per l'OpenGL.
- El projecte Mobles: És on l'usuari pot parametritzar una sèrie de mobles que després podrà afegir a l'habitació.
- El projecte Vista 3D: És on es fa la Visita Interactiva, és a dir, on l'usuari pot "caminar" per la l'habitació.
- El projecte Vistes: Ens mostra diferents perspectives de l'habitació en una mateixa finestra.
- El projecte Vista Prèvia: És el més senzill de tots, el vam separar en un projecte a part per a poder executar-lo des de diferents llocs, ja que la seva funció és mostrar la textura que s'ha seleccionat en qualsevol dels projectes anteriors.

Per a l'execució dels projectes des de la base de dades no hi havia problema, ja que existeix amb Visual C++ l'ordre ShellExecute, on li passem la ruta del projecte que volem i ens l'executa.
El problema era de comunicació entre ells, a vegades no n'hi havia prou amb només executar. Per exemple, per obrir i reeditar l'habitació d'un client, es pot fer un ShellExecute de l'Editor d'Habitacions, però com sap aquest quin fitxer ha d'obrir? Com distingir-los? El projecte Base de Dades no pot passar-li una variable a l'Editor de Mobles amb el nom del fitxer, aquest tipus de comunicació no existeix.

Si que es pot però vincular un arxiu d'un projecte a un altre, i això ens ha anat molt bé, perquè hem inclòs els fitxers de l'ADO[3] als projectes que necessitaven accedir a la base de dades. Però això no ens resol el problema comentat anteriorment.

La solució per la que vam optar va ser la utilització de fitxers. Un projecte escriuria en aquest fitxer la informació que després, de forma cronològica, obriria el programa que necessités la informació. Hem utilitzat sempre el mateix fitxer i s'anomena PLANTA_ACTUAL.TXT, ja que aquesta informació sempre és referent a una habitació (excepte el Vista Prèvia que utilitza el fitxer "nom_textura.txt").

Seguint amb l'exemple, quan obrim una habitació escollint-la a la llista i premem sobre acceptar, el projecte Base de Dades obre el fitxer i hi escriu a dins el nom de l'habitació. Quan l'Editor d'Habitacions s'executa, el primer que fa és obrir el PLANTA_ACTUAL.TXT i en llegeix el nom. Així ja sap quina és l'habitació que ha de carregar.

Aquest mètode és perfectament fiable i no produeix cap error, ja que el projecte està estructurat de forma que l'Editor de Mobles mai s'executarà sense que abans la Base de Dades hagi participat i hagi escrit la informació al fitxer.

El mateix passa amb la comunicació del projecte Dibuixar, Visita Interactiva i Vistes, i està resolt de la mateixa manera per tots i amb el mateix fitxer.

[3] II.2.- ADO pàg. 122

- PART I -
EL PROJECTE "DIBUIXAR"

I.1.-Explicació General

I.1.1.- Requeriment de l'aplicació "Dibuixar":

Al començar el programa vam començar a aprende les primeres aplicacions senzilles amb OpenGL. Al crear el nostre primer espai o món en 3D, vam veure que l'OpenGL llegia les superfícies d'aquest món amb una sèrie de punts lligats amb triangles formant una espècie de mapa de coordenades, també anomenat malla 3D. Per tant abans de posar-nos a editar l'habitació amb OpenGL necessitàvem tenir aquesta malla, ja que la gràcia era que l'habitació fos diferent segons la personalització del client.

Necessitàvem que d'alguna manera el client pogués dibuixar de manera ràpida, eficient i senzilla la seva habitació i que automàticament es creés aquesta malla.

Si tenim en compte que el programa es basa en el disseny interior d'una habitació, la millor manera de crear aquesta habitació és que pugui ser fins i tot el mateix client qui se la dibuixi de manera totalment personalitzada, tant en el que fa les parets com les portes, les finestres, les textures, el terra, el sostre, ...

La solució més efectiva era crear una aplicació de dibuix, aparentment senzilla, on de manera molt gràfica, utilitzant el ratolí i amb tota llibertat de moviments, es pogués dibuixar l'habitació sense restriccions.

A més l'aplicació havia de crear de forma amagada a l'usuari la malla 3D, ja que té un format massa complicat. La solució era que l'aplicació agafés el dibuix de l'usuari i el convertís a un format que l'OpenGL pogués interpretar.

Per tant, l'aplicació té dos grans objectius. Per una part ha de ser un editor en 2D fàcil d'utilitzar i sense restriccions, permetent dibuixar qualsevol tipus d'habitació. I per l'altra part havia de ser capaç d'agafar aquest dibuix i traduir-lo per a les llibreries OpenGL.

Passem ara a explicar de forma molt resumida i sense tecnicismes el funcionament de cadscun d'aquests dos objectius, en apartats posteriors ja profunditzarem en la part tècnica i de codi.

Editor 2D i Fitxer de Sortida.

I.1.2.- 1er Objectiu: Editor 2D

L'Editor 2D és la part de l'aplicació que està en contacte amb l'usuari, per tant s'ha de mirar de crear un mecanisme que sigui visualment fàcil d'entendre i a l'hora sigui eficient.

En aquest primer objectiu necessitem dibuixar línies, pintar de diferents colors, dibuixar rectangles,... per tant hem de fer una mena de programa per a dibuixar de l'estil del PaintBrush, on amb unes poques clicades de ratolí puguis fer un dibuix de les parets d'una habitació.

Així doncs, la idea és que la interfície tingui una zona delimitada per a dibuixar on l'usuari vagi creant les parets de l'habitació clic a clic. I amb unes quantes eines pugui dibuixar portes, parets, finestres, esborrar...

La primera idea per a fer l'habitació era de dibuixar-la amb línies ortogonals, és a dir amb línies únicament horitzontals i verticals, perquè així els punts sempre treballen només amb un sol eix de coordenades, cosa que simplifica molt les coses. Però fent-ho així teníem el problema evident de que perdíem molta llibertat, ja que no totes les habitacions estan fetes d'angles rectes.

La idea és que amb el ratolí marquem els vèrtex de cada paret (o porta, o finestra) de la vista en planta, i per a que les parets s'ajustin, els vèrtex seran enters i a més utilitzarem una funció que ens ajudi a corregir els errors a l'hora de clicar. D'aquesta manera aconseguim un dibuix esquemàtic de l'habitació que ja interpretarem.

Per a que l'habitació guanyi realisme en la part 3D i l'usuari es pugui fer una millor idea de com li quedarà la seva habitació amb els mobles posats, aquest ha de poder triar quina textura li va bé en cada superfície.

Per aconseguir tot això i simplificar al màxim la feina de l'usuari hem creat una sola interfície amb la zona delimitada per dibuixar, un botó per afegir portes, un per afegir finestres, un botó per tirar endarrera (desfer) i un altre per esborra-ho tot.

L'usuari ha de poder escollir les textures que vulgui, aquestes no tindran efecte fins a la part de "Visualització, Edició i Navegació".

Volíem posar-les en una llista per a que l'usuari les seleccionés. Això plantejava un problema, si volíem posar les textures en llistes ens sortia una llista per les textures de les parets, una per les de les portes, una altra per les de les finestres, una altra pel terra i una última per les textures del sostre, en total 5 llistes, no ens quedava espai en la interfície. La solució ha sigut posar una sola llista que tingui per defecte les textures de les parets i que vagi canviant el contingut segons vulguis afegir una porta o una finestra, ... D'aquesta manera hem optimitzat l'espai, podent fer més gran la zona per a dibuixar, i a més, així aconseguim fer-li fer a l'usuari les coses pas per pas i així evitem errors.

També se li demana a l'usuari que indiqui quina alçada té la paret.

Cal dir però, que la funció més important que fa aquest editor és guardar cada punt del pla (x,y) on l'usuari fa un clic per allargar la paret en unes llistes que més endavant la part del Fitxer de Sortida farà servir.

Al pròxim punt explicarem de manera més específica i més tècnica aquesta part.

I.1.3.- 2on Objectiu: Fitxer de Sortida

El segon objectiu, no entra en contacte amb l'usuari, sinó que es tracta de trobar una manera de poder comunicar a la part OpenGL el que ha fet l'usuari en la part de l'Editor 2D.

Hem de mirar de respectar al màxim el que ha dibuixat l'usuari, això vol dir que hem d'agafar el màxim d'informació possible de la part de l'Editor 2D.

Primer de tot cal tenir clar que les llibreries d'OpenGL llegeixen vèrtex i aquests els pot interpretar com a triangles o rectangles. En el nostre cas li passarem triangles, ja que a base de triangles podem dibuixar-ho pràcticament tot. D'aquesta manera per a construir una paret recta només ens fan falta 2 triangles ajuntats de manera que formin un sol rectangle.

Si tenim en compte que l'Editor 2D ens guarda cada punt que fem, si anem agafant els punts de 2 en 2 i sabem l'alçada de la paret podrem dibuixar 2 triangles que formin un rectangle.

Aquests és el mètode: Tenim els tots els punts que necessitem per a fer les parets del pla (x,y), que a l'OpenGL serà el (x,z), i amb l'alçada que fa la paret tenim el punt (y). D'aquesta manera passem de 2 dimensions a 3 dimensions, de (x,z) a (x,y,z). Si anem repetint aquesta operació tindrem tota la paret amb 3D i punt per punt.

També cal agafar les textures que ha triat l'usuari..

La idea principal és crear un arxiu de text (.txt) on hi hagi tota aquesta informació ben estructurada (s'ha de pensar que cal tractar diferent les portes i les finestres que les parets, per exemple) per tal que l'OpenGL el pugui llegir i interpretar directament i generar l'habitació en 3D.

Al pròxim punt explicarem de manera més específica i més tècnica aquesta part.

I.2.-Explicació tècnica de l'aplicació "Dibuixar"

Creiem que la millor manera d'explicar tècnicament l'aplicació Dibuixar és dividir-ho amb 3 apartats, un que explica la zona de dibuix que és la part essencial de l'aplicació, una altre que explica com s'afegeixen les portes i les finestres i un últim que diu com es tracten totes les dades obtingudes en els dos primers apartats i es passen al fitxer txt de sortirda.

I.2.1.- La zona de dibuix:

Estructurem aquest apartat en sis punts:

Delimitació i conversió de punts:

Per a fer un editor en 2D d'estil PaintBrush el primer que es necessita és una zona que quedi delimitada clarament per a que l'usuari pugui dibuixar. Això ho aconseguim fent servir una eina del *Microsoft Visual C++* que crea un rectangle de diferent color al de l'interfície en general.

Tenint la superfície de dibuix delimitada i tenint en compte que dibuixarem amb el ratolí cal diferenciar quan el punter es trobi dintre de la zona i fora de la zona. Per això fem servir una funció anomenada "dintre", a la qual li entren les coordenades del punt on es troba el ratolí i calcula si es troba dintre de la zona o fora:

- Si és fora, al marge superior esquerra de la pantalla surt un missatge que diu: "Estàs fora del rang de dibuix." i el punter del ratolí és el normal (IDC_ARROW). A més, es desactiven totes les funcions que permeten dibuixar, ja siguin parets, portes o finestres.
- Si és dintre, al marge superior esquerra surten les coordenades actuals del punter del ratolí, que a més passa a ser un creureta (IDC_CROSS). Prepara el programa per a començar a dibuixar i si ja s'ha dibuixat algun punt, mostra la distància de l'últim punt creat al punter del ratolí.

En quant a les coordenades, s'han de tenir en compte dos aspectes. Un aspecte és que quant necessitem agafar el punt de la pantalla on es troba el punter del ratolí, cridem a la classe CPoint de les llibreries de *Microsoft Visual C++* per a que ens capturi les coordenades on es troba, però aquestes coordenades s'han de convertir ja que l'origen el

situa a la part superior esquerra de la pantalla i a nosaltres ens interessa que l'origen de coordenades es trobi en el centre de la zona de dibuix.
El segon aspecte que hem de tenir en compte és que les mesures en que treballa OpenGL són força més petites que les que treballa *Microsoft Visual C++*, per tant, per molt que es pugui allunyar el punt de vista amb OpenGL[4] igualment crearia figures molt grans que farien anar més lentes les aplicacions gràfiques de que disposa el programa. I també hi ha el problema de que l'eix de coordenades està invertit.

La solució a aquests dos aspectes és fer una conversió de les coordenades que ens dóna la classe CPoint de manera que siguin més petites pel que fa a la part OpenGL i fer una altra conversió per a que l'origen de coordenades se situi al centre de la zona per a dibuixar. S'ha de crear una conversió per les coordenades de l'eix X i una altra per les coordenades de l'eix Y:

$$X=(X/5)-127$$
$$Y= -((Y/5)-76)$$

- Dividint per 5 aconseguim fer les coordenades més petites i adquar-les a un tamany més normal per a l'OpenGL.
- Restant 127 i 76 fem que es centri l'origen de coordenades al mig de la zona per a dibuixar.
- Posant el negatiu a la Y l'eix de coordenades deixa d'estar invertit.

També cal dir que les mesures que hem calculat respecte les proporcions, ens surt que 1 metre equival a 1'78 punts de l'OpenGL.

$$1m = 1'78$$

Llista de Textures:

La llista de textures és el llistat des d'on l'usuari sel·lecciona la textura que vol aplicar sobre la superfície que representarà, en la part de visualització en 3D, la línia que està a punt de dibuixar.

Només hi ha una llista de texutres, i tenint en compte que tenim diferents tipus de textures haurem d'anar carregant i esborrant el contingut de la Llista de Textures. Per a fer-ho hem creat una funció "mostrar_text" que li entra una String amb el tipus i retorna

[4] II.2.- Introducció a OpenGL pàg. 46

tota la llista de textures referents a aquest tipus. Per tant és el tipus el que fa variar el llistat.
Els tipus poden ser:

- Paret: és el tipus que surt per defecte.
- Porta: només es llista al prémer el botó "Afegir Porta".
- Finestra: només es llista al prémer el botó "Afegir Finestra".
- Terra: es carrega al finalitzar el dibuix i prémer el botó "Següent".
- Sostre: es llista al haver sel·leccionat la textura del pel terra, entrat l'alçada de la paret i prémer el botó "Següent".

Així doncs, a la funció se li entra un tipus (depenent de l'estat en que es trobi l'aplicació), es connecta a la base de dades, recorre la taula Textura buscant que el tipus coincideixi, i si coincideix l'afegeix a la llista. Arribat al final retonra aquesta llista que conté totes les textures del tipus entrat.

En el moment de fer els punts la textura sel·leccionada es guarda en una taula que té com a índex la paret a la que va destinada, és a dir, no un punt sinó dos.

Dibuixar línies(Parets) i aproximació:

Un cop s'entra a l'aplicació, tenim les textures carregades, el ratolí dintre de la zona de dibuix i una textura sel·leccionada. Ja podem començar a dibuixar la paret. Cal tenir present que el programa tracta diferent el primer punt que marquem que tots els següents que fem fins que aixequem el llapis (botó dret del ratolí), ja que el que ens interessa és saber quantes línies hi ha i no quants punts hi ha.
Per això disposem de dos tipus de contadors globals, el "num_punts_comensa", que és el conta els primers punts de cada sèrie de rectes i el "num_punts" que és el que conta la resta. Així que sumats tindrem el nombre total de punts. D'aquesta manera cada vegada que s'afegeixi una porta, una finestra o s'aixequi el llapis haurem de tractar-ho com a primer punt (num_punts_comensa).

El primer punt es dibuixa al clicar amb el botó esquerra del ratolí a sobre qualsevol lloc de la zona delimitada per a dibuixar. Guardem les coordenades convertides en 2 llistes de punts, la de les X i la de les Y. Després, al moure el ratolí es crea una línia de color vermell que va des del punt clicat fins a la posició del ratolí, d'aquesta manera es té una referència de per on passarà la paret si fem el segon punt allí. A més, l'aplicació ens indica la longitud real d'aquesta recta, així podem calcular amb exactitud la llargada de

la paret que estem dibuixant. Si se surt fora de la zona de dibuixar deixarà de dibuixar-se.

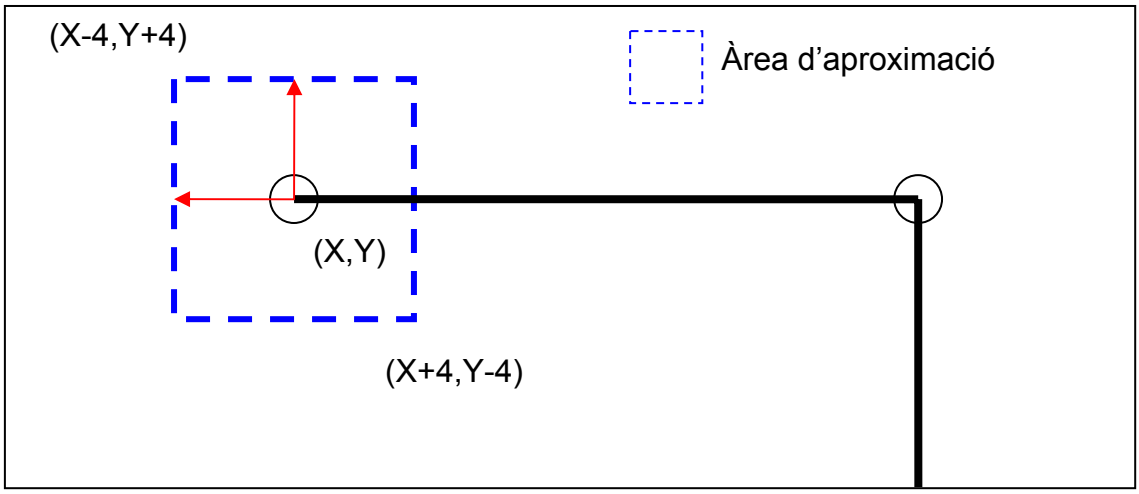
Cliquem amb el botó esquerra del ratolí i ja tenim el segon punt. El guardem a les llistes de punts, i al tenir dos punts concurrents ja podem dibuixar la línia que els uneix, i com que es tracta d'una paret serà de color negre. Cal dir que les línies les dibuixem gràcies a la classe CPen de la llibreria de *Microsoft Visual C++*. Creant un tipus de llapis, que varia segons el que hagi de dibuixar i amb les funcions MoveTo(punt2) i LineTo(punt1).

Per a seguir fent la paret només cal anar repetint la operació passant el que era el punt2 al punt1 i tornar a capturar la posició del següent clic. I fent aquests passos successivament tindrem tota la paret de l'habitació.

Si es vol fer trencar la sèrie de parets perquè es vol dibuixar una columna o una paret que està separada de la resta,... apretem el botó dret del ratolí i això crida a una funció que afegeix una línia d'avís a les llistes de punts (----). Així quan el programa ho llegeixi interpretarà que s'ha fet un salt i que s'ha començat a dibuixar en un altre lloc. A més, la línia vermella no sortirà i el pròxim punt que es dibuixi s'haurà de tractar com a primer punt (num_punts_comensa) .

Al voler tancar una sèrie de parets, unint l'últim punt amb el primer teniem la complicació de que no sempre encertavem el punt correcte i després l'habitació quedava oberta amb una escletxa en una cantonada.

Per a solucionar aquest problema ens vam haver de fer la funció "Aproximació", funció que se li entra el punt on es troba el ratolí i et retorna un punt, que pot ser el del mateix ratolí si no està a prop d'algun punt o retorna un punt que té molt proper. Per a fer-ho la funció recorre les llistes de punts, saltant-se els que no són enters (Porta, Finestra, ----), i si el punt en que es troba el ratolí està dintre d'un rang de +/- 4 tant en les X com en les Y respecte algun punt retorna aquest punt com a resposta. Si no, és a dir, si la distància del punt del ratolí al qualsevol dels punts que tenim guardats és més gran que 4 o més petit que -4, retorna el punt del ratolí:



Aproximació a un punt

L'aproximació s'executa mentre el ratolí es mou i estem dibuixant parets.

Redibuixar i la línia vermella:

La línia vermella ens va portar complicacions al començament, ja que és només temporal, al tornar a moure el ratolí de lloc la línia haurà de desaparèixer i dibuixar-se al següent lloc, agafant com a Punt2 el nou "punt" on hi haurà el ratolí. Primer vam intentar solucionar-ho dibuixant una línia blanca a sobre just en el moment de moure's i aprofitant que el fons de la zona de dibuixar és també blanca. Però això ens va portar a un altre problema: per allí on interseccionaven la línia vermella i qualsevol altra línia "s'esborrava", ja que al fer la línia blanca a sobre de l'altra ja dibuixada ho pintava com el fons. Així que vam acabar fent el Redibuixar.

El que fa el Redibuixar és situar-se a l'inici de les 2 llistes de punts i les recorre punt per punt, interpretant si hi ha portes, finestres o bé salts (aixecar llapis). Redibuixant les parets de color negre, les portes de color blau i les finestres de color verd.

Tot i així seguiem tenint el problema de que a vegades es quedava la línia vermella per sobre de la negra, de manera que vam acabar solucionant el problema fent que el Redibuixar abans de llegir a les llistes de punts dibuixés un rectangle blanc (amb la classe CRect) de les mateixes dimensions que la zona per a dibuixar.

El Redibuixar s'executa cada vegada que el ratolí es mou i estiguem en l'estat de "dibuixant". També s'executa quant surt un missatge d'informació.

El Desfer i el Borrar Tot:

Són dos processos que serveien per a modificar el contingut creat fins al moment. Tant l'un com a l'altre s'executen a partir d'uns botons situats en la barra d'eines de que disposa la interfície.

El Borrar Tot, com el seu nom indica esborra tot el que s'ha fet i et deixa en l'estat inicial. Destrueix tot el contingut de les llistes de punts, carrega a la llista de textures les textures que fan referència a la paret, inicialitza les variables tal i com estan a l'obrir l'aplicació i dibuixa un rectangle blanc a sobre de la zona per a dibuixar, de manera que la interfície queda tal i com es mostra a l'iniciar el Dibuixar.

I el Desfer elimini l'últim punt fet, així que si al fer un punt d'una paret veus que no ha sortit com esperaves, prems el botó i l'esborra tant de les llistes de punts com de la zona de dibuixar. I la línia vermella tindrà l'origen doncs en el penúltim punt guardat, que passarà a ser l'últim punt.

Aquest procés es pot anar repetint fins que no quedi cap punt guardat, en aquest cas, la finció seguirà el mateix procés que la funció Borrar Tot.

I.2.2.-Les Portes i Finestres:

Per tal d'aconseguir més realisme en les habitacions creades, havíem de poder afegir aquests dos elements, no només perquè existeixin a la gran majoria d'habitacions, sinó perquè també tenen una gran importància de cara a la col·locació dels mobles.

Vam plantejar la possibilitat d'afegir-les amb l'editor de mobles, però ho vam descartar perquè pensem que van massa lligades amb lo que és l'estructura de l'habitació, és a dir, que igual que amb les parets, un cop construïdes ja no es poden moure o "reeditar" com es pot fer amb els mobles. A més, afegint les portes i les finestres a la part del Dibuixar, permetíem també una millor precisió alhora de col·locar-les correctament a les parets.

Se'ns plantejaven una sèrie de problemes. Havíem de tenir en compte que una porta no té l'alçada de la paret, ni una finestra comença a terra i acaba al sostre, per tant, s'havien de tractar diferent a les parets. Tant les portes com les finestres s'havien d'ajustar obligatòriament arran de les parets, no podien quedar soltes. Un cop arran de paret, sorgia un altre problema: a quin costat de paret? És a dir, no es podia posar per exemple una finestra arran de paret però que aquesta fos per la part de fora, per tant, dins l'habitació ja no es veuria.

Com solucionar aquests problemes?
Per el de l'altura on arriba la porta i l'alçada on comença la finestra, vam decidir tractar-ho en un principi igual que les parets, quan en dibuixem la recta que representa la porta o la finestra, els dos vèrtexs d'aquesta s'escriuen a les llistes de punts, amb la diferència que abans d'escriure'ls a dins hi escriu o "Porta" o "Finestra". Per diferenciar-les de les parets, quan generem el fitxer de sortida l'aplicació llegirà en algun moment de les llistes de punts la paraula "Porta" o "Finestra" i reconeixerà que els dos punts següents no són parets. Més endavant, explicarem com es genera el fitxer de sortida i entrarem a comentar més específicament com tractem els punts de les portes i finestres[5].

Passem ara a explicar els dos problemes que quedaven estructurats en dues parts:

- L'Aproximació a Rectes i
- La Línia Doble

[5] II.3.7.- Fitxer de Sortida pàg. 105

- L'Aproximació a Rectes

No podíem permetre que l'usuari pogués dibuixar una porta o una finestra a qualsevol lloc de l'habitació, sense estar ben enganxada a la paret o senzillament col·locada pel mig de forma "flotant". En un principi ho arreglàvem amb un missatge per pantalla demanant a l'usuari que dibuixés una línia representativa de la porta o finestra arran de la paret, però havíem de restringir-ho de forma que s'hagués de fer obligatòriament, a més, per més que s'esforcés l'usuari a ajustar la recta arran de paret, mai quedaria dibuixat de la forma més precisa possible.

La solució és que quan es prem el botó "Afegir Porta" o "Afegir Finestra" la zona de dibuix fa com un canvi d'estat i mentre el ratolí es mou, en ves de fer una aproximació a un punt, explicada anteriorment[6], ha de reconèixer si es troba a prop d'una recta, reconeixent si es troba o no dins el radi d'aproximació d'alguna de les rectes del dibuix. Aquí ens trobem amb que hem de diferenciar entre dos tipus de recta, unes serien les ortogonals, les altres les diagonals.

Per a les ortogonals, no ens calen definicions de recta, utilitzem els dos punts que la formen. Senzillament li hem de definir un rang a x o a y, depenent de si la recta és vertical (x és la mateixa en tota la recta) o si és horitzontal (en aquest cas és y la mateixa). Parlem d'un cas vertical, on x és la mateixa a tota la recta de punt a punt: posem un rang de x a –x, quan el punt del ratolí s'aproxima fins al rang de x definit, ens iguala la variable x del ratolí a la x dels dos punts que formen la recta i se'ns enganxa de forma perpendicular ja que la y del ratolí es conserva.
Es dibuixa un punt blau que podem anar desplaçant al damunt de la recta ortogonal i ja estem llestos per dibuixar la porta o finestra.

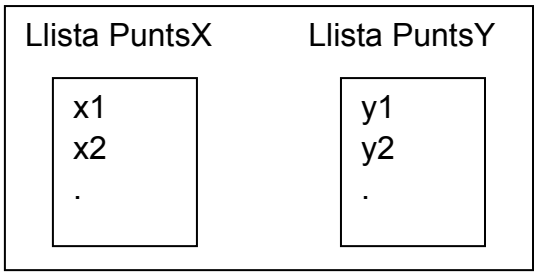
[6] I.2.2.- Portes i Finestres pàg 23.

En el cas de les rectes horitzontals, l'aproximació es complica, necessitem treballar amb rectes i per tant, les hem de definir.

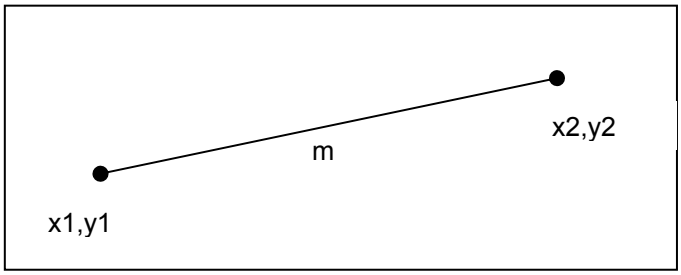
Necessitàvem reconèixer dins el dibuix, totes les rectes existents. També s'ha de posar al voltant d'aquestes rectes un radi d'aproximació per quan s'acosti el ratolí.
Tot això ho hem de fer cada vegada que el ratolí es mou (va canviant el punt d'aproximació), i per aconseguir-ho llegim tota la llista de punts agrupant-los pels que formen rectes entre ells i trobant-ne la pendent (m):

$$m = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

Fòrmula de la pendent



Representació de les llistes de punts



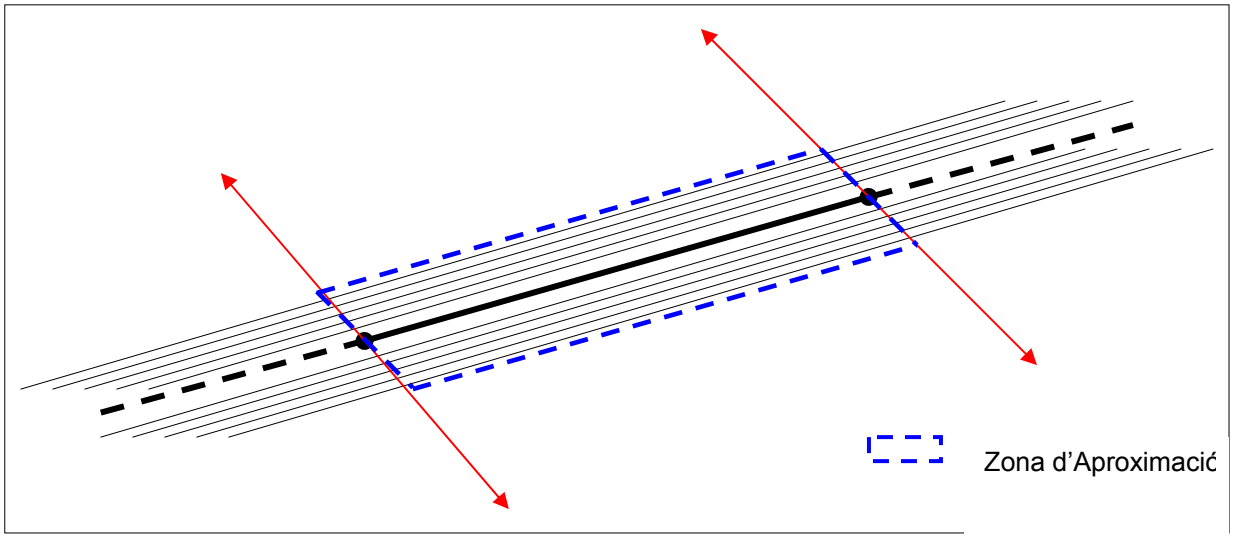
Representació de la recta definida

Ara, utilitzant l'equació de la recta, podem trobar el paràmetre "n", que representa el coeficient de posició de la recta:

$$y = mx + n$$

Equació de la recta

Finalment, sabem que dues rectes són paral·leles si i només si tenen les pendents iguals i els coeficients de posició diferents, per tant, ara podrem donar un rang d'aproximació a cada recta utilitzant la n (al projecte n<7 i n>-7). Calcularem si el punt on es troba el ratolí pertany a una recta paral·lela que està definida dins el rang de cada una de les rectes. A més també hem de mirar els límits per cada costat de la recta, sinó passant amb el ratolí per el costat de la recta ens faria igualment l'aproximació. Expressat gràficament:



Representació del rang d'una Aproximació a una recta

Quan el ratolí entra a la zona d'Aproximació, el que fem és agafar-ne el punt, i amb la pendent de la recta on s'ha aproximat, trobem la recta perpendicular entre aquesta recta i el punt del ratolí. Per trobar la pendent de la perpendicular:

$$m\_perp = -1/m$$

Pendent de la perpendicular

Ara tenim la m de la perpendicular (m_per), en podem calcular la n (n_per) ja que tenim també el punt (el del ratolí) i fem la relació:
Punt x_nou = (n_per-n) / (m-m_per);
Punt y nou = (m_per*x_nou) - (m_per*(punt x vell)+(punt y vell))
I ja tenim un punt blau dibuixat que podem anar desplaçant al damunt de la recta diagonal, llestos per dibuixar la porta o finestra[7].

- La Línia Doble

Un cop tenim feta l'aproximació, l'usuari veu un punt blau col·locat al damunt de la recta. Quan es troba en el punt desitjat de la recta, fa un clic al ratolí i el primer punt de la porta o finestra queda fixat i només pot anar allargant la línia per sobre de la recta.

Mentre s'estén la línia blava, pot anar mirant a la part superior de la interfície la mesura en metres que li està donant, de forma que ho pot ajustar tal i com ho té realment a la seva habitació.

El problema real que se'ns plantejava era què passava un cop feia el segon clic. No podíem clavar la porta o finestra exactament a sobre de la recta, ja que després, amb la visualització amb 3D, es creen problemes de textures, la textura de la paret i la textura de la porta queden sobreposades.

Una solució era desplaçar la recta de la porta o finestra cap a un costat de la paret, així quedaven clavades a la paret, de forma bastant precisa, i sense crear el problema de sobreposar les textures. Però, com reconeixeria l'aplicació si clavava la porta dins o fora l'habitació? Havíem de fer-ho diferent.

El millor era dibuixar dues línies, una a cada costat de paret. Aconseguíem dues avantatges, una que resolíem de forma definitiva el problema d'enganxar la porta, i l'altra, que des del punt de vista realista, no queda malament que en la visió en 3D veiem les portes i finestres enganxades dins i fora de l'habitació, com és realment.

Passem ara a explicar com fem l'extensió de la línia i després la creació de les dues línies enganxades. Cal a dir que també hem de diferenciar entre línies ortogonals i diagonals.

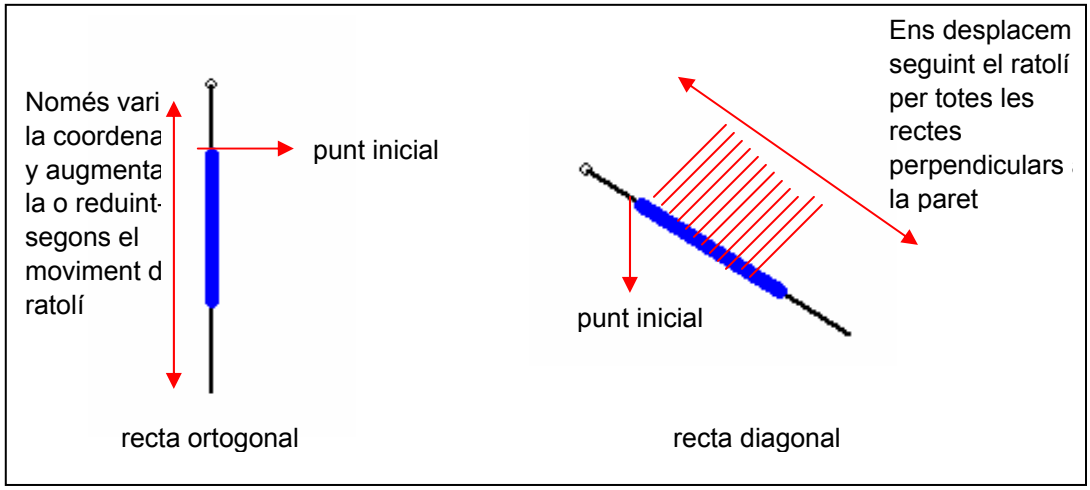
[7] I.2.2.- Portes i Finestres pàg 23

Amb les ortogonals, un cop dibuixat el primer punt, fem que l'usuari només pugui dibuixar canviant l'eix variable. En el cas d'una recta vertical, la línia només es va estenent variant la y (que ens la marca el punt on es troba el ratolí), de forma que la porta només es pot estendre amunt o avall (vertical) per sobre la recta.

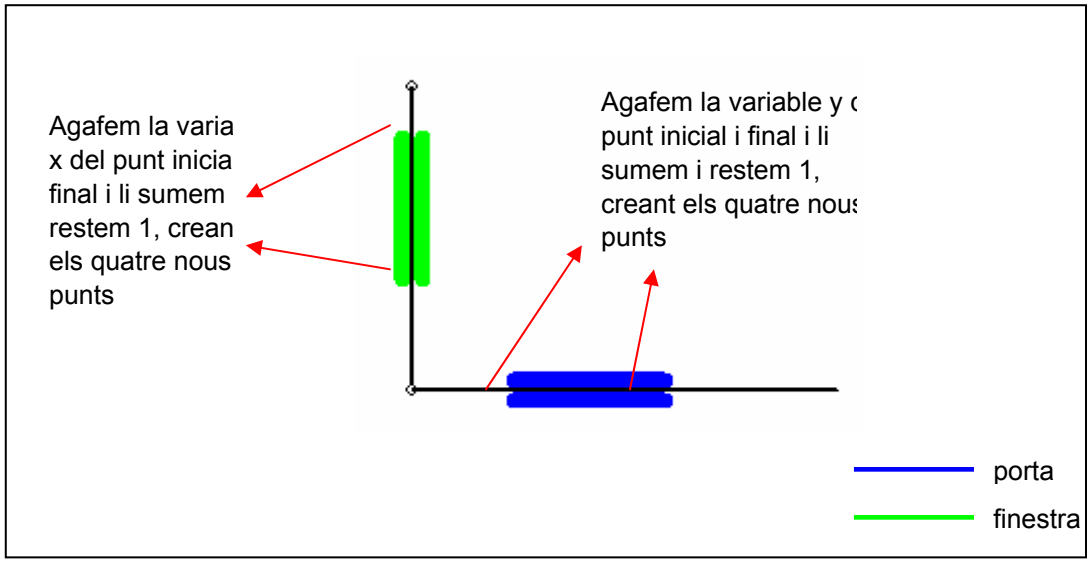
Amb les diagonals, el que fem és un cop fet el primer punt, anem estenent la porta per totes les rectes perpendiculars de la paret, utilitzant les fórmules de la pendent i perpendiculars explicada anteriorment.

Ara ja tenim els dos punts de la recta que representa la porta o finestra, hem de convertir aquesta recta en dues rectes, enganxades en cada costat de la paret. Per aconseguir-ho l'aplicació reconeix si la recta és vertical, horitzontal o diagonal, comparant els dos punts que formen la recta. Si és per exemple vertical, agafem els dos punts de la recta de la porta i hi apliquem els canvis (sumant i restant) a la variable x, de forma que convertim aquests dos punts en quatre, que formaran les dues rectes.
Si la porta és diagonal aplica els canvis tant a la x com a la y, i també ens genera quatre punts a partir dels dos que ja teníem.

Exemple Gràficament:



Desplaçament de la porta o finestra per sobre la paret



La doble línia

I.2.3.- El Fitxer de Sortida:

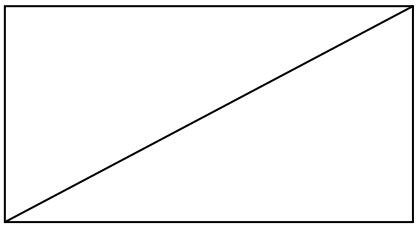
S'ha de crear un fitxer de sortida amb tota la informació obtinguda en les llistes de punts, la taula de textures, etc...

Per a començar-lo a fer cal prémer el botó "Següent", i com que encara ens falta informació per acabar de tenir l'estructura completa es demana ara. Necessitem saber l'alçada de la paret, la textura del terra i la textura del sostre. Per tant al prémer "Següent" es carrega a la llista de textures les que fan referència al terra i s'activa l'edit box que demana l'alçada. Una vegada sel·leccionada una textura de terra i entrada l'alçada de la paret es torna a prémer el botó "Següent" i aquest cop es carregarà el llistat de textures referents al sostre. Al sel·leccionar una de les textures es torna a prémer el "Següent" i ara sí ja a tenim tota la informació que necessitem per a crear el fitxer amb extenció .txt que llegiran les aplicacions OpenGL més endavant.

Primer, agafem el nom de l'habitació i creem el txt a partir d'aquest nom, així podrem relacionar la informació de l'habitació amb el fitxer que conté el codi per a l'OpenGL.

Necessitem saber el nombre de parets que hi ha en l'habitació. Així doncs haurem de mirar quins són els punts estan seguits i a quins hi ha salts per a poder saber el número de parets que hi ha, ja que cada paret la formen 2 punts seguits. Això ho aconseguim gràcies a diferenciar la veriable "num_punts_comensa", que són els punts per on es comença a dibuixar una paret, i la variable "num_punts" que són els punts que ens interessa saber, ja que equivalen al número de parets que hi ha.

Cal recordar que l'OpenGL funciona bàsicament amb triangles, i cada paret la forment 2 triangles com es mostra en la següent figura, així doncs haurem de saber més concretament quants triangles tenim per a crear un bucle per a omplir el txt amb tots els punts necessaris per a que sigui interpretat correctament.



Paret dividida en 2 triangles.

Per saber el número de triangles que tindrem només hem de mirar el "num_punts" i multiplicar-lo per 2, però a més hem de tenir en compte que ni el terra ni el sostre estan dintre de la llista de punts, per tant els haurem de sumar a part:

Número_Triangles=num_punts*2+2(sostre)+2(terra)

Un cop tenim el nombre de triangles ens disposem a començar a omplir el txt. Començarem posant el nombre d'Objectes que té l'habitació, és a dir 0, ja que encara no li hem pogut afegir-ne cap, pero ho necessitem per a poder modificar després. Seguirem escribint el nombre de triangles (polígons) tal i com hem calculat abans. Després serà el torn de l'alçada de la paret i finalment escriurem l'estructura que es repetirà contínuament:

-Tipus de superfície amb ID(Paret 5, Finestra 15, Terra 3,...)

-Identificador de la textura (textura 7, textura 25,...)

-Coordenades del triangle
Coordenades de la textura

-Identificador de la textura

-Coordenades del triangle
Coordenades de la textura

Així doncs creem el bucle que es repetirà tantes vegades com triangles tinguem.
Per a escriure els triangles començarem escrivint el Terra. Aquest es troba consultant tots els punts de les llistes i agafant els punts màxims i els punts mínims, així sempre quedarà quadrat i totes les parets quedaran per dintre.

Les textures omplen la superfície repetint-se un nombre determinat de vegades. Aquest nombre el treiem de calcular la llargada de la superfície en píxels i dividint-lo pel tamany de la textura, que sempre és de 128 píxels.

Seguirem ara ja amb les parets, les portes i les finestres. Aquí es llegeix de la llista de punts i poden passar 3 coses:

- Mentre no trobi "Porta", "Finestra" o "----" agafa dos punts, un serà per les coordenades *(x,z)* d'un dels punts i l'atre serà per la *(x,z)* de l'altre. El tercer punt l'aconseguirem al aixecar un o dos d'aquests punts amb l'ajuda de a coordenada *y*, que dependrà de si el punt del triangle toca al terra o té alçada i l'aconseguim de l'alçada que se li ha concedit a la paret. Després cal introduïr les coordenades de la textura, que les calculem com hem dit abans.
- Si troba "Porta" se salta el punt, però al crear els punts dels triangles té en compte que és una porta i per tant l'alçada no és tant alta com la de la paret i a més no repeteix la textura, ja que volem que la textura es vegi de dalta a baix de la superfície que representa la porta.
- Si troba "Finestra" actua de la mateixa manera que amb la porta, pero amb la diferència que l'alçada més baixa no serà 0 sinó que serà una mica més alta, ja que les finestres normalement no toquen al terra.
- Si troba "----" salta el punt i tracta en el següent punt torna a començar el procés.

En resum, l'estructura d'un triangle seria la següent:

(x1, y1, z1)(u_m,v_m)
(x2, y2, z2)(u_m,v_m)
(x3, y3, z3)(u_m,v_m)

On x, y i z són les coordenades del triangle i u_m, v_ m són les coordenades de la textura.

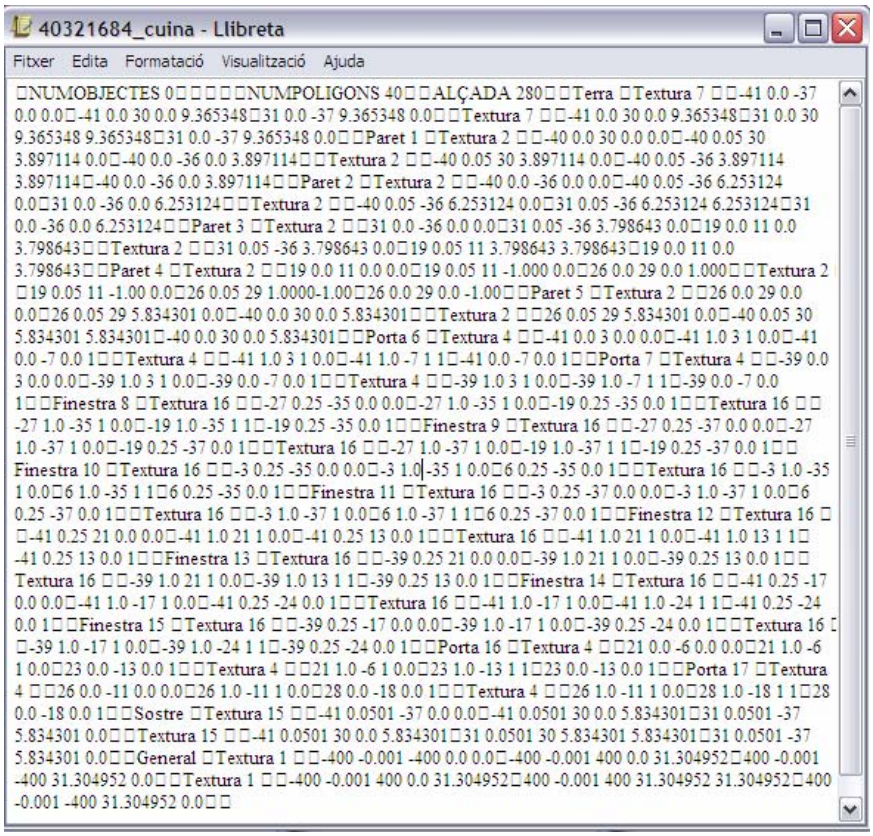
Finalment fem el Sostre, que serà com el terra, pero amb la mateixa alçada que la paret. I també creem el que en diem el Terra General, que només serveix per l'aplicació "Visita Interactiva"[8]

Abans de tancar l'aplicació necessitem 3 arxius diferens amb el mateix format, ja que més endavant els haurem de fer servir. Un per tenir l'habitació original, que per molt que li afegeixis mobles no varïi, un altre per l'Editor d'Habitacions i pel Vistes, que

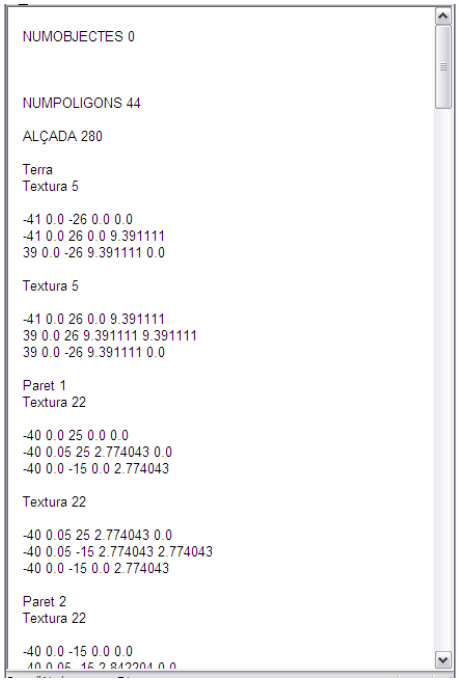
[8] II.4.- Visita Interactiva pàg 111.

s'hi puguin afegir mobles. I l'últim pel visita interactiva, que té un funcionament diferent al dels altres 2. Així doncs, agafem l'arxiu que hem creat i li fer 2 còpies exactament iguals i amb els seus noms corresponents per a no confondre'ls.

Aquí presentem un exemple d'un fitxer de sortida .txt:



Fitxer de Sortida complet.



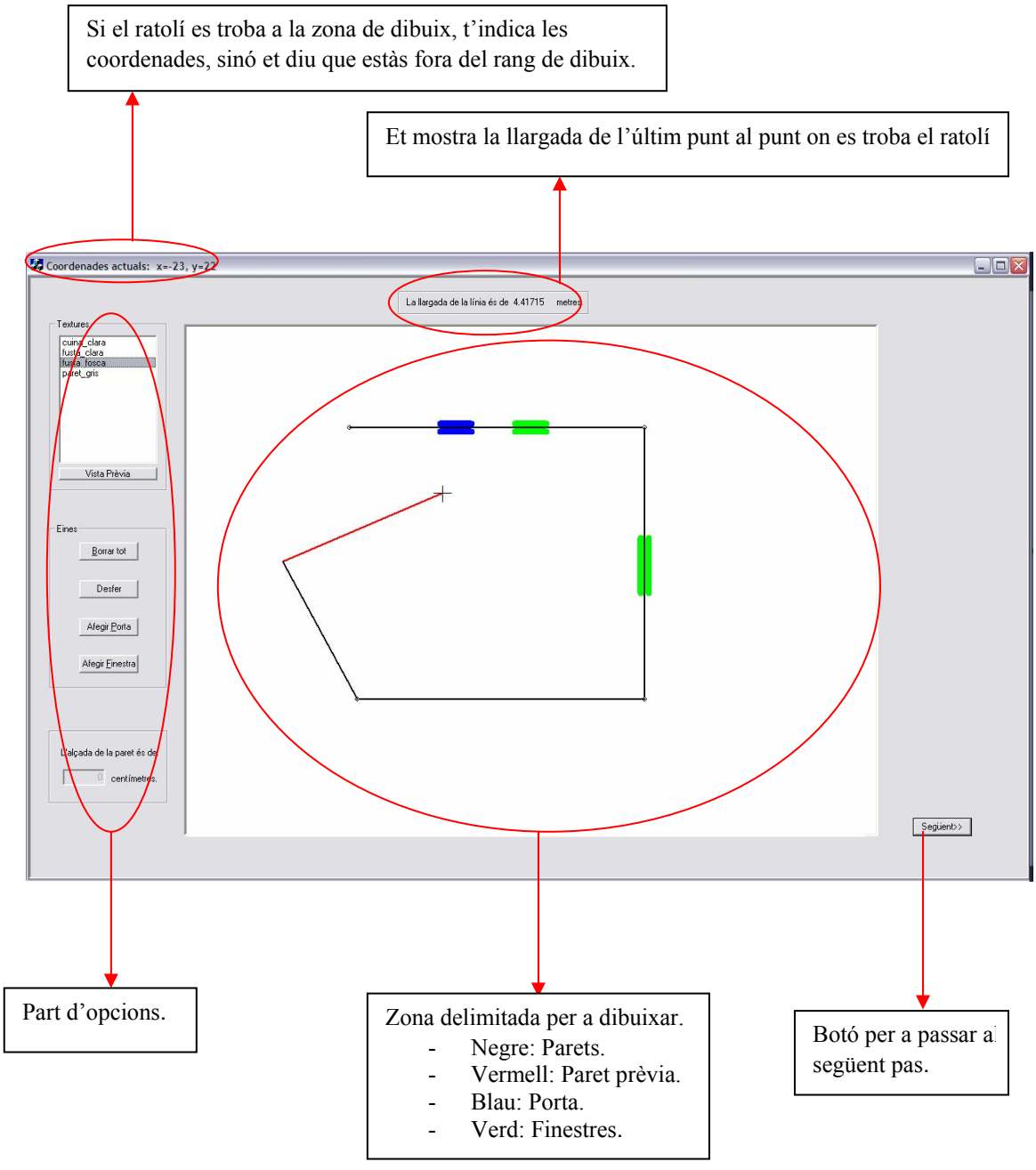
Detall del Fitxer de Sortida.

I.3.-Disseny i Anàlisi de la Interfície

I.3.1.- Explicació de la Interfície:

La interfície d'aquesta aplicació consta de 2 parts diferenciables, una la zona on es permet dibuixar i l'altra la zona que no, que és on hi ha tots els botons d'opcions, la llista amb les textures i l'edit de l'alçada.

-Interfície general:



-Part d'opcions:

1.-Llista

En aquesta imatege hi tenim la llista on es mostren totes les texutes que pot triar l'usuari.

Depenent de si es vol afegir una paret, una porta, una finestra, el terra o el sostre, la llista es carregarà amb unes textures o unes altres.

Per tant aquesta llista està plenament lligada a la base de dades i cada vegada que l'usuari vol afegir una cosa diferent s'hi connecta i hi fa una consulta (que més endavant veurem).

També hi ha un accés al projecte Vista Prèvia, que mostrarà la textura sel·leccionada en una altra interfície per a que l'usuari es pugui fer un idea més precisa del tipus de textura que ha sel·leccionat.

2.- Eines per a dibuixar

Hem intentat simplificar al màxim aquest apartat per mirar de fer-lo molt senzill, i això sumat a la gran dificultat que ens ofereix el *Microsoft Visual C++ 6.0* per a crear botons amb imatges, la nostra barra d'eines és així.

Consta de 4 botons:

Borrar tot: Eina que, com el seu nom indica esborra tot el que s'ha dibuixat fins al moment. Inicialitzant totes les variables i totes les taules com si entressim de nou a l'aplicació. A més redibuixa la zona de dibuixar a blanc i carrega la llista de textures amb les textures de paret.

Desfer: Aquesta eina, el que fa es eliminar l'última acció que s'ha fet. I ho fa restant 1 al número de punts i passant el penúltim punt fet a les variables que guarden l'últim punt fet.

Si es dóna el cas de que al apretar "Desfer" el número de punts és igual a 0, fa exactament com el "Borrar tot" i torna a inicialitzar les variables que ha fet servir.

Afegir Porta: Aquest botó "canvia d'estat", és a dir, el que fa és que deixa d'afegir parets i passa a afegir una porta, canviant les textures de la llista, i dintre de la zona per a dibuixar, el ratolí no s'aproxima als punts, com ho fa quan afegeix parets, sinó que s'aproxima a les rectes (buscant paral·leles).

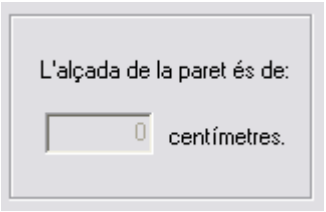
Abans d'escriure el punt a la llista de punts, indica que és una Porta escribint "Porta" i per qüestions que més endavant explicarem crea una altra porta paral·lela a la primera per a que hi hagi una porta a cada costat de la paret.

Quan es dibuixa la porta en la zona de dibuix queda marcada de color blau.

Afegir Finestra: El funcionament és similar, per no dir igual, que el del "Afegir Porta". La única diferència destacable és que en la zona de dibuixar surt de color verd i no de color blau com la porta.

A més, evidenment, la finestra té una alçada inferior més gran que la porta, i cal recordar que cap de les dos tenen una alça més gran que la paret, sempre serà inferior.

3.- Alçada de la Paret:



Aquesta imatge representa l'edit encarregat de guardar l'alçada de la paret. S'entra tot just al final, juntament amb la textura del terra i la textura de sostre, i és quan l'usuari ha decidit que ja ha acabat el dibuix, amb totes les finestres i portes que necessita, apreta el botó "Següent" i llavors s'activa l'edit box aquest per a poder entrar l'alçada, al mateix temps que canvia la llista de textures i es llisten les textures del terra. Al tornar a apretar "Següent" l'alçada queda enregistrada i a la llista hi surten les textures del sostre. Finalment tornem a apretar "Següent" i es crea l'arxiu .txt que necessitem per a que les llibreries de OpenGL ho puguin interpretar i es crei l'habitació que es preten.

I.3.2.- Llistat d'esdeveniments:

1. ***Nom esdeveniment:*** Es vol donar d'alta una habitació.
   Resposta: S'obre la interfícies de l'aplicació.
   Descripció: S'obre la interfícies de l'aplicació i es carrega en la llista de textures les textures referents a les parets. S'inicialitzant totes les variables.

2. ***Nom esdeveniment:*** Es vol dibuixar una paret.
   Resposta: Es dibuixa una línia i es guarden els 2 punts d'aquesta.
   Descripció: Primer es comprova que s'hagi sel·leccionat una textura, després mira si es tracta del primer punt de la paret o el segon, ja que pot ser que s'hagi "Aixecat el llapis" o bé que comencem a dibuixar des del principi.
   - Si és el primer punt, el guarda i espera a que es torni a clicar per a guardar el segon punt. Per a tenir una referència es dibuixa una línia vermella des del primer punt al punt on hi ha actualment el ratolí. Aquesta línia vermella es va esborrant i tornan-se a redibuixar a mesura que el ratolí es mou per la panatalla.
   - Si és el segon punt el guarda en una llista i després redibuixa l'anterior línia vermella de color negre.

3. ***Nom esdeveniment:*** Es mou el ratolí per la pantalla (sense res dibuixat).
   Resposta: Comprovar que el ratolí estigui dintre de la zona de dibuix.
   Descripció: Si el ratolí es troba dintre de la zona destinada a dibuixar al marge superior esquerra de la pantalla et mostra les coordenades en que es troba el punter, si no, és a dir, si es troba fora de la zona per a dibuixar com que no ens interessen les coordenades surt un text dient: "Està fora del rang de dibuix.".

4. ***Nom esdeveniment:*** Es mou el ratolí per la zona de dibuix (volem dibuixant parets).
   Resposta: Mostrar coordenades del ratolí, si és segon punt, indicar llargada de la paret i fer aproximació als punts ja fets.
   Descripció: Marca les coordenades on es troba el punter del ratolí, a cada moviment redibuixa la zona de dibuix i tornem a dependre de si és el primer punt o no:
   - Si és primer punt i no hi ha cap paret dibuixada només mostra les coordenades, pero si ja hi ha alguna paret dibuixada, al apropar-se a algun punt ja fet fa un salt des de la posició actual del punter a la posició del punt que s'ha aproximat, eina molt útil per acabar de dibuixar una habitació.
   - Si és el segon punt va dibuixant una línia vermella des del punt anterior al punt acutal del punter del ratolí. I també fa l'aproximació en cas d'estar a prop d'algun punt, sempre i quant no sigui el punt anterior, ja que després no es dibuixaria cap paret.

5. ***Nom esdeveniment:*** Es vol afegir una porta.
   Resposta: Carregar llistat de textures de porta i dibuixar la porta.
   Descripció: Al clicar el botó "Afegir Porta" es carrega la llista de textures amb les que corresponen a la porta. Aquest esdeveniment també consta de dos part:
   - Si és el primer punt de la porta: Comprova que s'hagi sel·leccionat alguna textura referent a la porta, comprova que en el moment de clicar hi hagi el punter estigui bastant a prop d'alguna paret ja feta, + mirar Esdeveniment 6 + afegeix a la llista de punts la paraula "Porta" per a que després el programa sàpiga que no es tracta d'una paret, sinó que ha de tractar els següents punts com una porta. Tot seguit guarda les coordenades del primer punt.
   - Si és el segon punt de la porta: Mirar Esdeveniment 6 + afegeix les coodenades del segon punt i a més a més, com que s'han de dibuixar 2 portes, una a cada costat de paret torna a omplir la llista de punts amb la paraula "Porta" i afegeix 2 coordenades més, una per a cada punt de la segona porta. Aquests punts els troba a partir de calcular una paral·lela de la primera porta. Finalment afegeix una línia de "----" que l'aplicació interpreta com que s'ha "Aixecat el llapis". S'ha de puntualitzar que per diferenciar les portes de les parets pintem les portes de color blau. Quan acaba, redibuixa la zona de dibuix.

6. ***Nom esdeveniment:*** Es mou el ratolí per la zona de dibuix (volem dibuixar una porta).
   Resposta: Comprovar que el ratolí estigui en una paral·lela d'una paret ja dibuixada.
   Descripció: Es va redibuixant la zona de dibuix. I també depen de si ja hem fet un primer punt de la porta o no:
   - Si encara no hem fet cap punt farem servir l'"Aproximació Paret", que és una funció que busca les paral·leles que estan a prop de la paret més propera al punter del ratolí, d'aquesta manera sabrem en quina paret l'usuari vol posar la porta. I es dibuixarà un punt blau, que anirà recorrent la paret més propera, indicant on aniria el primer punt de la porta en cas de clicar.
   - Si ja s'ha fet el primer punt, el que fa l'aplicació és allargar el punt blau tant com es mogui el ratolí de manera perpendicuar a la paret que s'ha sel·leccionat, per exemple si volem fer una porta en una paret horitzontal, per molt que pugem o baixem el ratolí el punt es quedarà igual, en canvi si el movem d'esquerra a dreta veurem com el punt es transfoma en una línia blava que recorre la paret que s'ha triat. Es mostra també la llargada que tindrà la porta.

7. ***Nom esdeveniment:*** Es vol afegir una finestra.
   Resposta: Carregar llistat de textures de finestra i dibuixar la finestra.
   Descripció: El funcionament és molt similar al de l'afegir una porta (veure esdeveniment 5), les úniques coses que canvien són que el carregar la llista de textures es carreguen les que fan referència a la finestra, que al moment de clicar, en lloc d'escriure a la llista de punts "Porta" esciru "Finestra" i es dibuixa de color verd en lloc de color blau.

8. ***Nom esdeveniment:*** Es mou el ratolí per la zona de dibuix (volem dibuixar una finestra).
   Resposta: Comprovar que el ratolí estigui en una paral·lela d'una paret ja dibuixada.
   Descripció: El funcionament és el mateix que si volguessim dibuixar una porta (veure Esdeveniment 6), l'únic que canvia és que ho fa de color verd en comptes de fer-ho de color blau.

9. ***Nom esdeveniment:*** Es vol tirar endarrere (desfer).
Resposta: Esborrar últim punt creat i actualitzar variables.
Descripció: Primer comprova que hi hagi alguna cosa dibuixada, si hi ha alguna cosa destrueix l'últim punt creat i resta un a la veriable que compta els punts. Després es fa un "redibuixar" de la zona de dibuix i es mostra el dibuix sense l'últim punt, que ja hem desfet.
En el cas de que es vagi apretant el botó "Desfer" i s'arriba a l'últim punt fa el mateix que el botó "Borrar tot" (veure esdeveniment 10).

10. ***Nom esdeveniment:*** Es vol esborrar tot el que s'ha dibuixat.
Resposta: Comprovar que hi hagi alguna cosa per a esborrar, esborrar tot i actualitzar variables.
Descripció: Es comprova que s'hagi dibuixat alguna cosa prèviament, si és així, reseteja totes les llistes de punts, inicialitza totes les variables i carrega la llista de textures amb les textures de la paret. Després fa un "redibuixar" i mostra la interfície com si tornessim a començar.

11. ***Nom esdeveniment:*** Es vol veure la textura sel·leccionada.
Resposta: S'executa l'aplicació "Vista Prèvia".
Descripció: Es comprova que s'hagi sel·leccionat una textura i si és així s'executa l'aplicació "Vista Prèvia" que et mostra un cub en 3D amb la textura a cadascuna de les seves 6 cares.

11. ***Nom esdeveniment:*** Es vol "Aixecar el llapis" (Apretar botó dret sobre la zona de dibuix).
Resposta: Cridem a la funció "Aixeca_Llapis".
Descripció: Com que quan comences a dibuixar parets van totes lligades (a no ser que posis portes o finestres), si es vol dibuixar una columna o un apèndix de l'habitació, o una paret que no formi part de l'estructura bàsica de l'habitació, s'apreta el botó dret del ratolí i ja no dibuixa la línia vermella que ens indica que la paret anirà de l'últim punt al punt actual del punter del ratolí. D'aquesta manera es pot dibuixar una altra secció de parets. I el que fa l'aplicació es afegir a la llista de punts una línia de "------" per a que l'aplicació entengui de que en aquell punt s'ha "aixecat el llapis". I s'ha d'indicar que el que s'entenia com a primer punt (que és l'últim punt dibuixat) ja no ho és.

13. ***Nom esdeveniment:*** Es vol acabar de dibuixar.

Resposta: Es guarda la informació obtinguda al txt de sortida.

Descripció: Al prémer el botó "Següent" es carrega la llita de textures amb les textures referents al terra i demana que se'n sel·leccioni una. A més també s'activa l'edit box que ens indica l'alçada que tindran els parets, l'usuari l'ha d'omplir. Una vegada sel·leccionada la textura pel terra i introduïda l'alçada de la paret, s'ha de tornar a clicar el botó "Següent" i després es carrega la llista de textures amb les textures referents al sostre. Una vegada sel·leccionada la textura, es torna a prémer e botó "Següent" i llavors s'interpreta tota la informació que hem estat guardant i s'escriu als txt de sortida (més endavant explicarem com es fa).

- PART II -
VISUALITZACIÓ EDICIÓ
I
NAVEGACIÓ

II.1.-Explicació General

La part de Visualització, Edició i Navegació és en la que es basa el programa. És on es pot veure l'habitació que ha creat l'usuari en 3D, des de diferents perspectives i punts de vista, on es poden afegir mobles, editar-los i reeditar-los.

També s'ha de dir que ha estat la part més complicada de fer, ja que els nostres coneixements en el tema de gràfics en 3D eren pràcticament nuls i la informació que corre per internet no és tanta com s'espera, a part dels exemples més senzills.

Aquesta part conté 6 projectes:

- El projecte *Vista_Planta*: també l'anomenem Editor d'Habitacions, és el projecte més important de tots i és aquí on tractem l'habitació, on afegim els mobles i els editem.
- El projecte *Vista_3D*: també l'anomenem Visita Interactiva i és un projecte que ens mostra el resultat final del programa veient l'habitació des d'una prespectiva de primera persona, és a dir, l'usuari pot "caminar" per dintre de l'habitació que ha creat i ha amoblat.
- El projecte *Vistes*: es tracta d'una finestra que conté quatre subfinestres on es pot veure l'habitació des de 4 angles diferents i ajustables. Ideal per a imprimir i tenir una idea general de com ha quedat l'habitació.
- El projecte *Mobles*: també l'anomenem Mobles a Mida. És allí on hi tenim llistats els mobles que l'usuari pot crear a la seva mida. El projecte els crea a partir de les dades que ha entrat l'usuari i els prepara per a que es puguin afegir a l'Editor d'Habitacions.
- El projecte *Carregador_ASE*: també l'anomenem Mobles Stàndar. És semblant al projecte Mobles a Mida, pero els mobles en que treballa aquest projecte no són editables, tot i que es poden crear amb programes com el 3Dstudio Max, Autocat,... cosa que els hi dóna molt de realisme.
- El projecte *Vista Prèvia*: és el projecte més petit de tots i és que l'únic que fa és mostrar un cub pintat amb una textura prèviament sel·leccionada, i ens serveix per veure com són les textures abans d'aplicar-les.

Per a poder treballar amb gràfics havíem de triar entre les dues grans llibreries gràfiques diferents que existeixen per a *Windows*: *DirectX (Microsoft) i OpenGL(SGI)*. Finalment ens hem decantat per OpenGL.

Haguéssim pogut posar un apartat amb moltes pàgines d'informació tòrica sobre el que és OpenGL, pero hem preferit fer un petit apartat introductori i a mesura que anem explicant la part II explicarem el funcionament i la manera com hem utilitzat OpenGL.

II.2.- Introducció OpenGL

L'OpenGl és el lligam software per al hardware gràfic. És un motor 3D en el qual les seves rutines estan integrades a les targetes gràfiques.
Té totes les característiques necessàries per a la representació d'escenes 3D modelades amb polígons, des del pintat més bàsic de triangles, fins al mapeig de textures.

La companyia que va desenvolupar aquesta llibreria es Sillicon Graphics Inc (SGI), per tal de fer un estàndard en la representació 3D gratuït i amb codi obert (open source).

Actualment és una de les tecnologies més utilitzades en el disseny d'aplicacions 3D.
La podríem dividir en tres parts funcionals:

- La llibreria OpenGL, que proporciona tot el necessari per accedir a les funcions de dibuixat d'OpenGL.

- La llibreria GLU (OpenGL Utility Library), una llibreria d'utilitats que proporciona accés ràpid a algunes de las funcions més comuns d'OpenGL., a través de l'execució de comandes de més baix nivell, pertanyents a la llibreria OpenGL pròpiament dita.

- GLX (OpenGL Extension to the X Window System) proporciona un accés a OpenGL per poder interactuar amb un sistema de finestres X Window, i està inclòs en la pròpia implementació d'OpenGL.

A més d'aquestes tres llibreries, la llibreria GLUT (OpenGL Utility Toolkit) proporciona un sistema independent de plataforma per crear aplicacions de finestres totalment portables [GLUT].

OpenGL com a màquina d'estats:

L'OpenGL treballa com una màquina d'estats. Quan s'activen o configuren varis estats de la màquina, els seus efectes perduren fins que siguin desactivats, realitzant certes acciones, que tindran com a fi una representació en pantalla (o no) d'una sèrie de dades, depenent de l'estat en què ens trobem...
Per exemple, si el color per pintar polígons es posa a blanc, tots els polígons es pintaran d'aquest color fins canviar l'estat d'aquesta variable.

Tots els estats tenen un valor per defecte, i també alguna funció amb la qual aconseguir el seu valor actual.

Amb l'OpenGL, l'ordre de les accions resulta crític en la majoria de les ocasions..., no serà el mateix traslladar i rotar, que rotar i traslladar, com veurem més endavant[9].

D'aquesta manera, i de forma molt abstracta, el procediment de dibuixar alguna cosa en OpenGL sol ser la següent:

1. Activar totes les opcions que seran persistents a l'escena (posem la camera, activem els filtres...).

2. Activar les opcions que estableixen l'estat d'un objecte específic (la seva posició en l'espai, la seva textura...).

3. Dibuixar l'objecte.

4. Desactivar les opcions pròpies d'aquest objecte (tornar a la posició original, desactivar la seva textura).

5. Tornar al punt 2 fins que haguem dibuixat tots els objectes.

[9] II.2.- Introducció a l'OpenGL pàg 46.

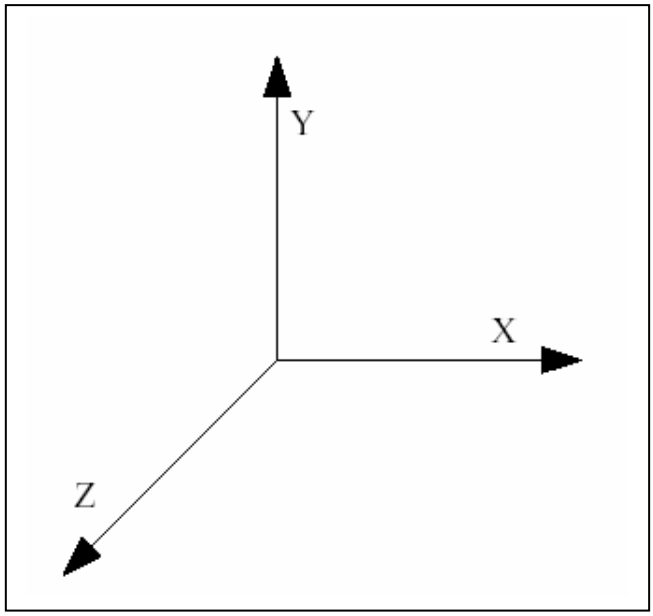
II.2.1.-Les Funcions bàsiques amb OpenGl

La sintaxi:

Totes les funcions de d'OpenGL comencen amb el prefix "gl" i les constants amb "GL_". Com a exemple, la funció glClearColor() i la constant GL_COLOR_BUFFER_BIT.
En el cas de les funcions GLUT, el prefix de les funcions és "glut", per exemple glutCreateWindow.
En moltes de les funciones, apareix un sufix compost per dos caràcters, una xifra i una lletra, com per exemple glColor3f() o glVertex3i(). La xifra simbolitza el nombre de paràmetres que se li han de passar a la funció, i la lletra el tipus d'aquests paràmetres.

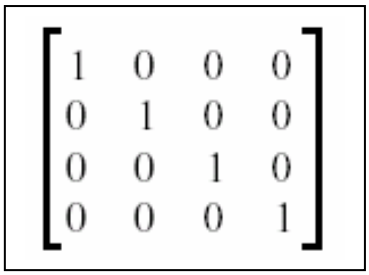
L'Espai 3D:

L'OpenGL proporciona accés a funcions per dibuixar 2D, però nosaltres al projecte treballem sempre en l'espai 3D. L'OpenGL treballa en principi en un espai de 3 dimensions, encara que més endavant veurem que en realitat treballa amb coordenades homogènies (4 dimensions). Las tres dimensions que ens interessen són les especificades per un sistema 3D ortonormal. És a dir, els seus eixos són perpendiculars, i cada unitat en un d'ells està representada per un vector de mòdul 1 (si ens allunyem una unitat, ens allunyem la mateixa distància del l'eix de coordenades, és igual la direcció).



Eixos de coordenades en OpenGl

D'aquesta manera, el sistema de coordenades inicial d'un sistema OpenGL pot representarse amb la següent matriu:



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriu de transformació

La situació dels eixos de coordenades es reflexa en la matriu de transformació. Aquesta matriu representa la transformació que s'aplicarà a tots els vèrtexs que es dibuixin mentre ella estigui activa.

Les Coordenades Homogènies:

Quan pensem en el dibuixat 3D, estem acostumats a pensar en espais de tres coordenades (o dos per figures 2D), però en OpenGL tot es tradueix a coordenades homogènies. Les raons són vàries, entre elles es guanya uniformitat d'operació amb matrius i la facilitat de diferents conceptes relatius a la profunditat.

D'aquesta manera, el punt 3D (1,2,3) és traduït per OpenGL al punt (1,2,3,1.0) i el punt 2D (1,2) és traduït a (1,2,0.0,1.0). Podem dir que un punt (x,y,z,w) en coordenades homogènies, és equivalent al punt 3D (x/w,y/w,z/w).
Així els punts (1,2,0,10), (1,2,0,1) i (1,2,0,0.00001) es corresponen amb els punts 2D (0.1,0.2), (1,2) i (10000,20000).

Un apunt interessant és que si la coordenada w és igual a 0, estariem parlant del punt situat a l'infinit en la direcció del vector que especifiquen les altres tres coordenades.

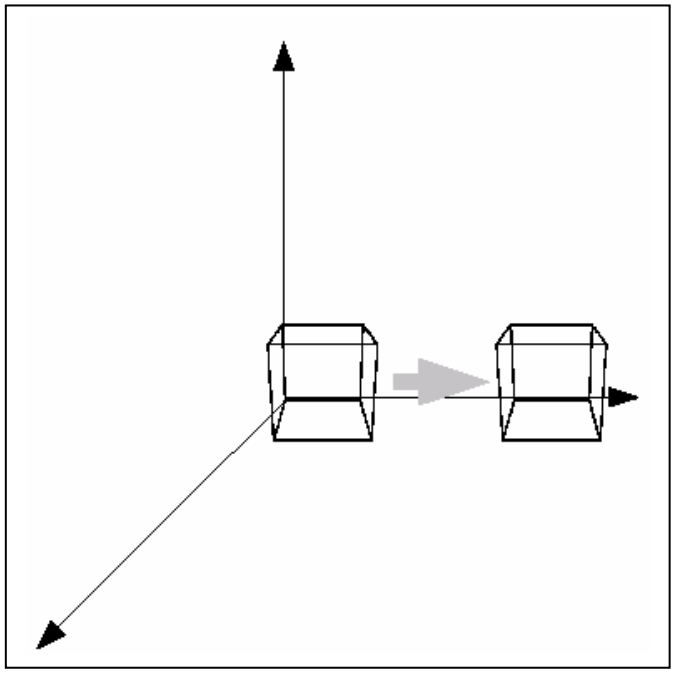
Activació i desactivació d'opcions:

Como ja hem comentat, l'OpenGL és una màquina d'estats: activem i desactivem opcions que afecten al dibuixat. Habitualment, les opcions s'activen i desactiven amb **glEnable** (<OPTION>) i **glDisable**(<OPTION>).
Per exemple glEnable(GL_TEXTURE_2D) activarà les textures.

Les transformacions d'objectes:

Aquestes transformacions són les que descriuran com es visualitza un objete en l'espai, al nostre projecte n'utilitzem de dos tipus:

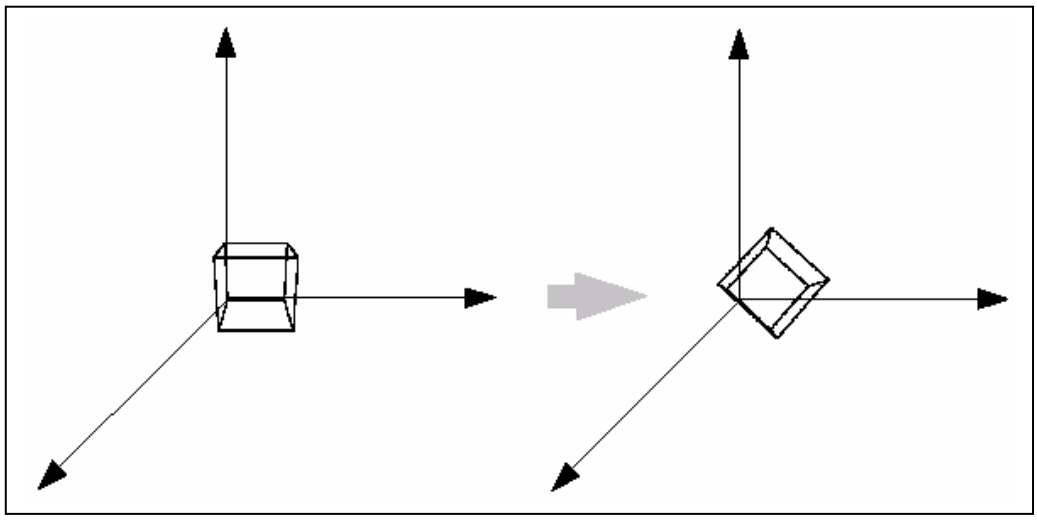
- Translació: una translació és un desplaçament d'un objecte en l'espai...Per exemple, movem un objecte a una nova posició des de l'actual.



Translació d'un objecte

La funció en OpenGL es diu:
- **glTranslate**: ens permetrà traslladar un objecte en l'espai.

- **Rotació:** com el seu nom indica, un objecte rota al voltant d'un eix que passa per el seu "centre de gir".



Rotació d'un objecte

La funció en OpenGL es diu:
- **glRotate**: ens permetrà rotar un objecte en l'espai.

Com funcionen aquestes transformacions?
Tota transformació que apliquem construeix una matriu de quatre dimensions que es multiplicarà per la matriu de transformació inicial (aquella que defineix l'eix de coordenades). Per exemple, al traslladar un objecte dos unitats en l'eix X, es genera la següent matriu de transformació:

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exemple matriu transformació

Si apliquem aquesta transformació a la matriu original, ens quedarà que la nova matriu de transformació es:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

matriu per matriu

Si ara dibuixem el punt (1,0,0), tenint en compte que la matriu indica un desplaçament de dos unitats en l'eix X, el punt hauria de dibuixar-se en la posició (3,0,0). Per això, es multiplica el punt per la matriu de transformació:

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

matriu per punt

Per poder multiplicar el punt (1,0,0) per la matriu, l'hem convertit a coordenades homogènies, i per a això, hem afegit una última component amb el valor 1.

Les Matrius:

Anteriorment hem vist que OpenGL guarda la transformació dels objectes en una matriu. A aquesta matriu se la denomina matriu de visualització/modelatge, perquè s'utilitza per a aquestes dues funcions.
A més d'aquesta transformació, OpenGL té una altra matriu molt important, que és la matriu de projecció, en la que es guarda la informació relativa a la "camera" a través de la qual veurem el mon.
Al realitzar operacions que modifiquen alguna d'aquestes dues matrius, haurem de canviar el "mode de matriu", perquè les operacions afectin a la matriu que ens interessa.
Per a això utilitzarem les funcions **glMatrixMode**(GL_MODELVIEW) o **glMatrixMode**(GL_PROJECTION).
També existeixen dues funcions que permeten guardar i restaurar els valors de la matriu activa en una pila.

La funció **glPushMatrix**() guarda una matriu al cim de la pila, i **glPopMatrix**() la treu, i la restaura. Això ho podem utilitzar per dibuixar un objecte i, abans de dibuixar el següent, restaurem la transformació inicial. Per exemple:

```
<transformació comú per l' escena>
glPushMatrix();
<trasformació pròpia de l'element 1>
<dibuixat de l'element 1>
glPopMatrix(); // tornem a la transformació comú
glPushMatrix();
<transformació pròpia de l'element 2>
<dibuixat de l'element 2>
glPopMatrix(); // tornem a la transformació comú
...
```

Finalment, comentar l'operació **glLoadIdentity**(), que carrega la matriu identitat com a matriu activa.

El dibuixat en OpenGL:

Per dibuixar en OpenGL, hem d'habilitar el mode de dibuixat, establir les opcions de dibuixat de cada vèrtex, i dibuixar-ne cada un d'ells. Al acabar de dibuixar una figura, finalitzem el mode de dibuixar.

Per començar a dibuixar, utilitzarem la comanda **glBegin**(CONSTANT MODE DE DIBUIXAR), on aquesta constant pot ser de tipus GL_POINTS, GL_LINES... En el nostre projecte dibuixem amb el mode GL_TRIANGLES, on cada tripleta de vèrtexs s'interpreta com un triangle.

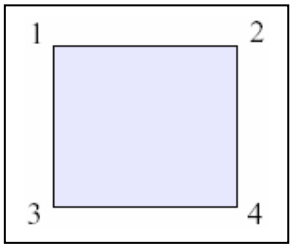
Entre les funcions que permeten establir els atributs de cada vèrtexs, estan aquelles que ens deixen seleccionar el seu color (glColor), normal (glNormal), coordenades de textura (glTexCoord), etc.
Finalment, les funcions de dibuixat de vèrtexs tenen la forma "glVertex".

El color:

OpenGL pot utilitzar dos mètodes de color: color RGBA i color indexat. Nosaltres ens centrem en el color RGBA. Es composa de quatre components: Vermell (Red), Verd (Green), Blau (Blue) i canal Alfa, o transparència.

La orientació de les cares:

Un polígon té dues cares, la de davant i la de darrera. La manera de saber quina cara és la de davant, i quina la de darrera, és que, si mirem la de davant, els vèrtexs s'hauran dibuixat en ordre antihorari.
Per exemple, tenim la següent figura:



orientació de les cares

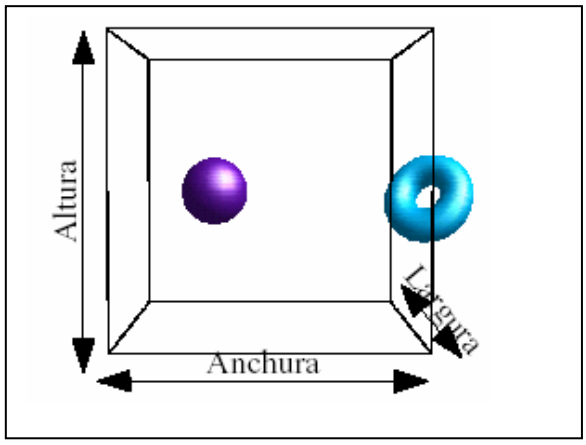
Si dibuixem els vèrtexs en l'ordre 1,3,4,2, estarem dibuixant la cara davantera mirant cap a nosaltres, però si l'ordre és, per exemple, 1,2,4,3, estarem mirant la cara de darrera del polígon.

La projecció en OpenGL:

En el mode de projecció podem especificar com afectarà la posició d'un objecte a la seva visualització. Tenim dues maneres de visualitzar l'espai:

- La projecció ortogràfica:

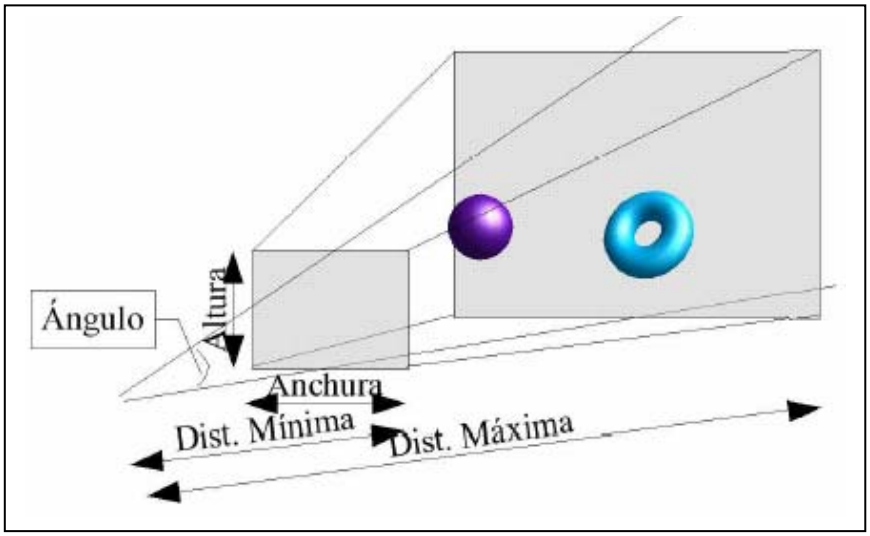
La projecció ortogràfica ens permet visualitzar tot aquell que es trobi dins d'un cub, delimitat per els paràmetres de la funció glOrto. A la hora de visualitzar, la distància a l'observador només es té en compte per determinar si l'objecte està dins o fora del cub...



projecció ortogràfica

- La projecció perspectiva:

La projecció perspectiva delimita un volum de visualització donada per un angle de camera, i una relació alt/ample. La distància a l'observador determinarà el tamany amb el què un objecte es visualitza.



projecció perspectiva

II.2.2.- El GLUT (OpenGL Utility Toolkit):

La llibreria glut està dissenyada per no tenir tantes preocupacions respecte al sistema de finestres, incloent funcions per crear-les independentment de la plataforma. Són funcions del tipus obre_finestra() (per exemple), que ens amaguen la complexitat de llibreries a més baix nivell.

A més, GLUT ens ofereix tota una dinàmica de programació d'aplicacions OpenGLl, gràcies a la definició de funcions callback. Una funció callback serà cridada cada vegada que es produeixi un event, com la pulsació d'una tecla, el reescalat de la finestra, etc. El callback passa un punter a una altra funció que serà cridada per la funció principal.

Quan utilitzem aquesta llibreria, li donem el control del flux del programa a GLUT, de forma que executarà codi de diferents funcions depenent de l'estat actual del programa (el ratolí canvia de posició, etc.).

II.3.- Editor d'Habitacions

II.3.1.- Introducció, requeriments i la Interfície:

L'Editor d'Habitacions és la part fonamental del programa, és on l'usuari veurà inicialment la seva habitació des d'una perspectiva de planta, tot i que la podrà veure des de qualsevol angle. També podrà afegir-hi els mobles que estiguin disponibles, desplaçar-los, elevar-los, rotar-los,...

Per a poder dur a terme tot això necessitarem dividir la finestra en vàries parts. Aquestes parts seran subfinestres GLUT, ja que la utilització d'aquesta llibreria, en aquest cas, et facilita molt les coses, perquè haurem de crear menús amb clics de ratolí, haurem de tenir un llistat de mobles on els puguem seleccionar per a després editar-los, i un tauler de control amb uns quants botons on al clicar-hi es realitzin accions.

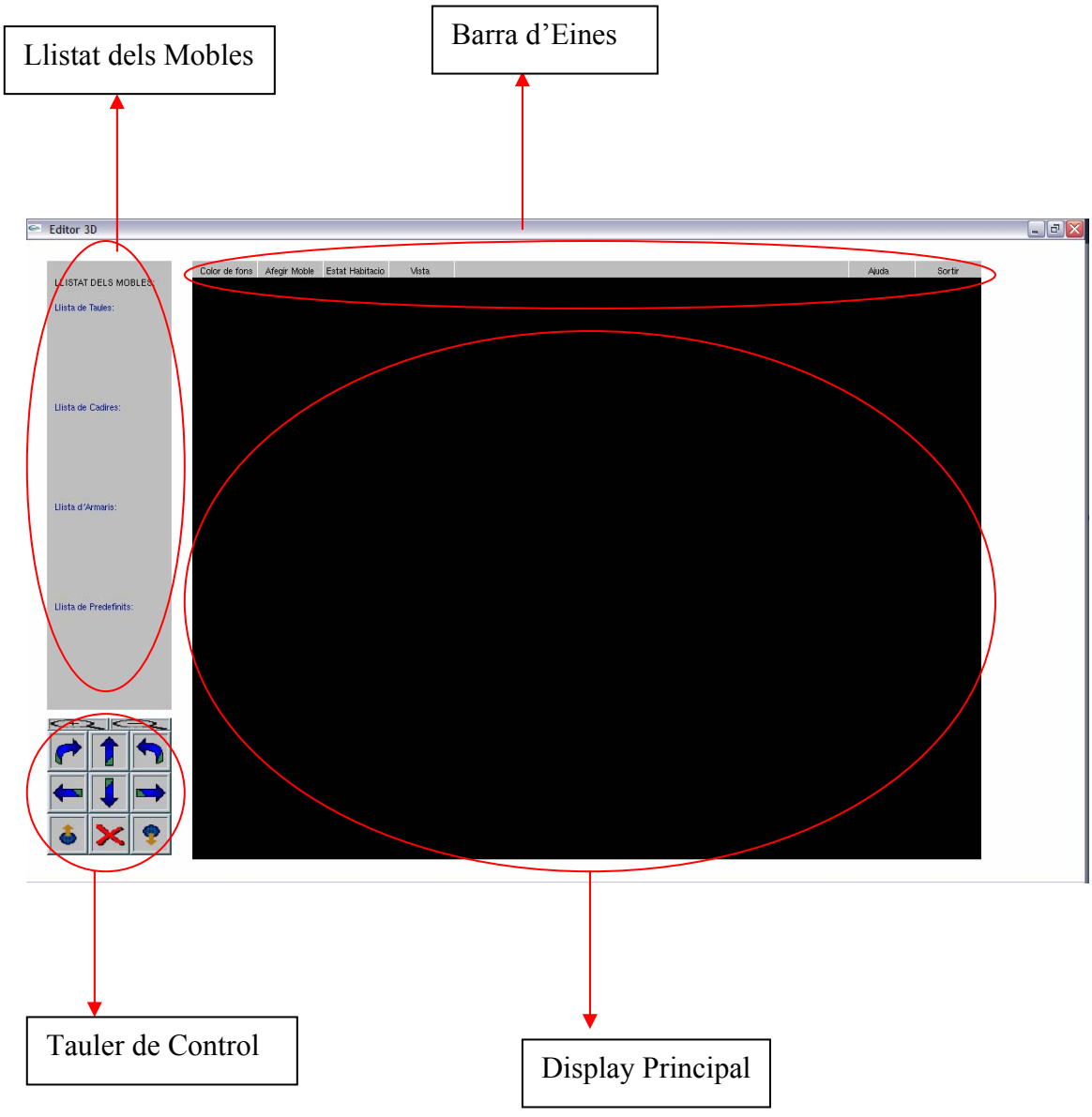
Cal dir que l'Editor d'Habitacions és la part que més ens ha costat i més temps hi hem dedicat. És la part que més treballa amb OpenGL i a més engloba altres parts molt importants del projecte com ara els mobles.

En un principi, la intenció era fer-ho tot en una sola finestra on al clicar el botó dret del ratolí t'apareixia un menú amb totes les funcions de l'Editor. Però de seguida vam veure que els requeriments eren molt més elevats.

En aquesta part del projecte és on hem anat aprenent OpenGL, primer crear les finestres, després crear l'escena principal amb l'habitació, després afegir objectes dins d'aquesta escena, etc.

La Interfície:

A l'executar l'Editor d'Habitacions veiem que se'ns obre una finestra amb diferents seccions.



La interfície de l'Editor està formada per una finesta GLUT gran i que engloba un conjut de subfinestres també GLUT. En OpenGL l'ús de les finestres és fonamental, ja que una finestra pot servir de tot: per a mostrar una figura en 3D, com a quadre de text, com a botó,...

Al executar l'Editor d'Habitacions es comença per la funció "main" que és l'encarregada d'organitzar les finestres i subfinestres en l'espai i li diu a cada una a quines funcions poden fer, aquestes funcions s'anomenen normalment callbacks i que són els esdeveniments que succeeixen sobre cada finestra.

Creació de finestra i subfinestra GLUT:

Per a crear una finestra GLUT primer ens posem en mode display i indiquem els tipus de buffers que farem servir i la gamma de colors. Després el tamany que tindrà la finestra i la posició que tindrà en l'espai.

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(1200, 750);
glutInitWindowPosition(0, 0);
```

Després li posem el nom que volguem que tingui i la creem. I finalment li associem els callbacks pel seu display, el seu teclat, el seu ratolí,... i també es poden afegir altres funcions normals que volguem que tingui en compte la finestra definida.

```
finestra = glutCreateWindow("Editor 3D");
        glutDisplayFunc(main_display);
        glutKeyboardFunc(teclatNormal);
        glutSpecialFunc(teclatEspecial);
        ...
```

I ja tenim la finestra creada. Per a crear un subfinestra GLUT el procediment és molt similar, pero partim de la base de que el *glutInitDisplay* ja està activat, i cal tenir en compte de que les subfinestres no porten títol. Primer associem la subfinestra a la finestra, en aquest cas es la subfinestra "planta" l'associem a la finestra "finestra", i li entrem les dimensions que ha de tenir. Després se li associen els callbacks, en aquest cas els de display, teclat normal, teclat especial, els clics del ratolí i el moviment del ratolí. I en aquest cas també li hem associat unes quantes funcions.

```
planta= glutCreateSubWindow(finestra, 200, 45, 950, 700);
        glEnable(GL_TEXTURE_2D);
        LoadGLTextures();
        glEnable(GL_DEPTH_TEST);
        glutDisplayFunc(display);
        glutKeyboardFunc(teclatNormal);
        glutSpecialFunc(teclatEspecial);
        glutMouseFunc(onMouse);
        glutMotionFunc(onMotion);
        ...
```

- glEnable(GL_TEXTURE_2D) és la funció que activa les textures bidimencionals per a que puguin ser carregades.
- LoadGLTextures() és l'encarregada de carregar les textures que tenim, ja l'esplicarem millor en el seu apartat[10].
- glEnable(GL_DEPTH_TEST), és la funció que activa el z-búffer o búffer de profunditat. En un principi OpenGL actua de manera que al llegir uns quants objectes dóna prioritat de visió a l'objecte que s'ha entrat més tard. Això crea el problema de que si per exemple si s'afegeixen una televisió i després una taula per aquesta televisió per més baixa que sigui la taula sempre es veurà per sobre de la televisió. La manera d'arreglar-ho és activant el z-búffer, que el que fa és donar prioritat de visió el triangle que està més aprop del punt de vista.
- Diferència entre glutKeyboardFunc i glutSpecialFunc: totes dues fan referència a esdeveniments de teclat, la primera es refereix a les tecles normals del teclat, és a dir les que surten en el codi ASCII i la segona es refereix a les tecles especials del teclat, com ara les fletxes, l'espai, l'enter, el tabulador,...

[10] II.3.2.2.- Carregar Textures pàg. 65.

Les seccions de la interfície:

- *Display Principal*: És una sola subfinestra i és la que ens mostrarà l'espai 3D que utlilitzem per a mostrar l'habitació, afegir els mobles,...
- *Barra d'Eines*: Està formada per 7 subfinestres que en aquest cas fan de menú d'opcions. És a dir, cada subfinestra fa en realitat de botó, que al clicar-hi a sobre desplega un menú GLUT
- *Llistat dels Mobles*: És la llista dels mobles de que disposa la planta. És una sola subfinestra dividida en 4 tipus de mobles que a la vegada estan dividits en 10 cel·les i en cada cel·la hi cap un moble.
- *Tauler de Control*: Són 11 subfinestres que funcionen com a botons. A cada subfinestra se li associa una textura representativa de la seva funció. L'objectiu que tenen és controlar el moviment que hi ha dintre del Display Principal.

Més endavant ja explicarem en més profunditat cadascuna d'aquestes seccions.

II.3.2.-Carregar Habitació:

Passem ara a explicar com carreguem l'habitació a la subfinestra Display Principal, és el moment de llegir el fitxer de sortida de la part del projecte Dibuixar[11]. De moment, i per explicar les coses pas a pas només comentarem com es carrega l'habitació buida, és a dir, sense cap moble. En apartats posteriors, explicarem com es carrega l'habitació si hi ha mobles per tal de reeditar-los[12].

Utilitzem dues funcions imprescindibles per aconseguir mostrar l'habitació, es diuen SetupWorld() i crea_mon(), considerem que són molt importants i per tant les explicarem de forma detallada[13]. Abans però hem d'explicar dos conceptes molt importants, els registres i el com es carreguen les textures.

II.3.2.1.-Els registres o "TAD's":

És imprescindible per al projecte el tema dels "TADs" o registres, que és informació que carreguem a la memòria RAM. Aquests estan estructurats de forma jeràrquica. Anem a veure'ls, del més petit al que els engloba a tots, amb el codi i tot:

El registre Vèrtexs:

```
typedef struct tagVERTEX
{
   float x, y, z;
   float u, v;
} VERTEX;
```

Aquí definim un vèrtex, que en realitat no és més que un punt. Li passem les coordenades del punt en 3D (x,y,z), i les coordenades de com assignarem la textura a aquest vèrtex (u,v).

[11] I.2.3.- Fitxer de Sortida pàg. 30
[12] II.3.4.- Mobles pàg. 79.
[13] II.3.2.3.- Dibuixar l'habitació pàg. 66.

El registre Triangle:

```
typedef struct tagTRIANGLE
{
   int tex;
   VERTEX vertex[3];
   int tipus;
} TRIANGLE;
```

Definim un triangle, veiem que engloba 3 registres del tipus vèrtexs, ja que un triangle són 3 vèrtexs. Per definir un triangle, a part dels 3 vèrtexs necessitem també saber quina textura té associada (int tex), i la posició d'aquesta textura ja la tenim definida a cada vèrtex en el registre. També definim un tipus de triangle (int tipus), que en realitat només tindrà valor "1" en el cas que es tracti d'un triangle que forma part del terra general[14] o del sostre, o "0" sinó es tracta de cap d'aquests dos tipus.

El registre Sector:

```
typedef struct tagSECTOR
{
   int numtriangles;
   TRIANGLE* triangle;
   int textura1, textura2, textura3,textura4, textura5, textura6, textura7,
textura8,  textura9, textura10,rotar, repetir;
   float despx,despy,despv,xpota,ypota,zpota,cub;
   int ID_ASE;
} SECTOR;
```

Un sector està determinat per un nombre variable de registres triangle (int numtriangles). També hi englobem tretze variables de tipus integer, deu per a textures, una per a la rotació, una per al nombre de repeticions i una altra per a l'identificador d'un moble de tipus ASE. Hi tenim també set variables de tipus float, que són les coordenades de desplaçament i el nombre de cubs que pot tenir un armari. Veiem que aquestes variables entraran en joc més endavant quan entrem a comentar els mobles.

[14] II.4.- Visita Interactiva pàg. 111.

El registre Mon:

```
typedef struct tagMON
{
   int numobjectes;
   SECTOR* sector;
} MON;

MON mon1;
MON mon_objecte;
```

El registre Mon ve definit per un nombre de registres sector (int numobjectes).
Definim dos tipus de Mon, el tipus mon1, que serà l'habitació en si, i el tipus mon_objecte, on cada sector serà un objecte.
Ara ens interessa doncs el mon1, que és l'habitació i que està formada per un sol sector.

II.3.2.2.-Carregar les Textures:

Passem a explicar com l'OpenGL carrega les imatges en BMP per convertir-les a textura. Utilitza la funció LoadGLTextures().

El primer que fa és una connexió a la base de dades, per veure el nombre de textures que tenim. En teoria, aquesta funció funciona posant-hi una llista de totes les textures.BMP, però nosaltres havíem de poder afegir textures, per tant, no podíem fer anar aquesta funció de forma constant, havia de poder anar canviant.

El que fem un cop sabem quantes textures tenim a la base de dades és un bucle on cada una de les textures que tenim retorna el seu nom de la imatge .BMP i amb això especifiquem la ruta on es troba. També retorna el seu ID, i com que les textures queden carregades a la memòria RAM, ara el projecte ja identificarà les textures amb el número que li passem. Després el propi OpenGL transforma aquestes imatges utilitzant una sèrie d'ordres:

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glBindTexture(GL_TEXTURE_2D, textures[loop]);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[loop]->sizeX,
TextureImage[loop]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE,
TextureImage[loop]->data);
```

D'aquestes ordres nosaltres n'especifiquem el GL_REPEAT, que significa que les textures s'aniran repetint més d'una vegada sobre la paret (per exemple), la resta de constants són referents a una sèrie de filtres perquè la imatge es vegi millor i per fer la conversió d'imatge a textura.

II.3.2.3.-Dibuixar l'Habitació:

Ara que tenim els registres i les textures explicades, podem comentar ja les funcions que ens carreguen una habitació. Comencem per la funció display() de la subfinestra Display Principal:

```
void display(void)
{
   float colors[6][3] =
   {
      { 0.00f, 0.00f, 0.00f}, // 0 - negro
      { 0.06f, 0.25f, 0.13f}, // 1 - verde oscuro
      { 0.10f, 0.07f, 0.33f}, // 2 - azul oscuro
      { 1.00f, 1.00f, 1.00f}, // 3 - blanco
      { 0.12f, 0.50f, 0.26f}, // 4 - verde claro
      { 0.20f, 0.14f, 0.66f}, // 5 - azul claro
    };
    glClearColor(colors[iFondo][0], colors[iFondo][1], colors[iFondo][2], 1.0f);
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(90.0f, 1.0f, 1.0f, 50.0f);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(0.0f, 0.0f, 12.0f,0.0f, 0.0f, 0.0f,  0.0f, 1.0f, 0.0f);
   glTranslatef(xp,yp,zp);
   glRotatef(alpha, 1.0f, 0.0f, 0.0f);
   glRotatef(beta, 0.0f, 1.0f, 0.0f);
   crea_mon();
   ...
   ...
   ...
```

El display, quan s'inicia veiem que el primer que fa és (a part de fixar el color de fons) fixar les matrius. Amb el glMatrixMode(GL_PROJECTION), fixem la perspectiva del que serà el dibuix de l'habitació (gluPerspective) que podríem definir com a la relació d'aspecte. Després definim el glMatrixMode(GL_MODELVIEW), on hi especifiquem el punt de vista (gluLookAt), la posició (glTranslatef) i l'angle (glRotatef) que tindrà l'habitació dins la finestra.

Un cop definides les matrius, carreguem la funció crea_mon(), que és qui ens dibuixa l'habitació cada vegada que s'executa la funció display().

Per entendre aquesta funció però, cal haver explicat la SetupWorld(), ja que és lo primer que fa la crea_mon(), per entendre-ho millor, la primera carrega l'habitació amb el fitxer, i la segona és qui la dibuixa.

La funció SetupWorld:

Aquesta funció és qui llegeix del fitxer.TXT totes les coordenades i informació de textures que hi tenim, i ho interpreta amb el procediment OpenGL per llegir malles 3D. Val a dir que carrega una habitació tant buida com amb mobles i que dibuixa l'habitació amb parets o sense segons se li demana però ara ens centrarem en carregar només l'habitació.

Primer de tot obre el fitxer corresponent a través del PLANTA_ACTUAL.TXT[15], un cop té el fitxer obert i en llegeix la primera línia opera de la forma següent:

```
readstr(filein,oneline);
sscanf(oneline, "NUMOBJECTES %d\n", &numobjectes);
 loop_p=0;
mon1.sector= new SECTOR[numobjectes];
mon1.numobjectes=numobjectes;
```

[15] Requeriments Tècnics pàg. 8.

El primer que es troba al fitxer és el nombre d'objectes que conté l'habitació i que en aquest cas serà 0, ja que ara només carrega l'habitació per primer cop. Afegim al registre mon1 un nou sector (serà el sector 0 en aquest cas) i li donem valor a la variable numobjectes també amb 0 en aquest cas. Fins aquí hem inicialitzat el registre mon1 de l'habitació.
Continua llegint el fitxer i es troba:

```
readstr(filein,oneline);
sscanf(oneline, "NUMPOLIGONS %d\n", &numtriangles);
readstr(filein,oneline);
sscanf(oneline, "ALÇADA %d\n", &altura);
mon1.sector[loop_p].triangle = new TRIANGLE[numtriangles];
mon1.sector[loop_p].numtriangles = numtriangles;
```

Agafa el nombre de triangles que té l'habitació (expressat en NUMPOLIGONS al fitxer) i seguidament l'alçada de les parets. Ara carrega al registre sector el nombre de triangles que ha llegit i dona valor a la variable numtriangles del sector.
Ara entra en un bucle que va del primer triangle a l'últim (numtriangles), i mentres va llegint les línies del fitxer, i per a cada triangle comprova si forma part del sostre o del terra general[16], si el triangle en forma part, dona valor a la variable tipus del triangle:

```
if ((sscanf(oneline, "Sostre \n", &alsada)) || (sscanf(oneline, "General \n", &alsada)))
{
   mon1.sector[loop_p].triangle[loop].tipus=1;
   mon1.sector[loop_p].triangle[loop+1].tipus=1;
}
else
mon1.sector[loop_p].triangle[loop].tipus=0;
mon1.sector[loop_p].triangle[loop+1].tipus=0;
```

Si formen part del sostre o del terra general, dóna valor 1 a la variable tipus del registre triangle, sinó li assigna 0 de valor. Això ho necessitem perquè el terra general i el sostre no apareixen a l'editor d'habitacions, per tant aquesta variable ens serà útil per després demanar-li quins triangles ha de dibuixar i quins no.

[16] II.4.-Visita Interactiva pàg. 111.

Recordem que estem dins un bucle que va recorrent la informació de cada triangle, fins aquí ja tenim carregats el registre mon1, el sector, dels triangles quants n'hi tenim i ara de cada un anem mirant de quin tipus són. Ens falten la variable text de cada triangle i les coordenades dels vèrtexs que el formen.

Ara mostro el format del TXT a mesura que es va trobant Parets:

Paret 1
Textura 0
-58 0.0 28 0.0 0.0
-58 0.05 28 3.281101 0.0
-58 0.0 -22 0.0 3.281101
Textura 0
-58 0.05 28 3.281101 0.0
-58 0.05 -22 3.281101 3.281101
-58 0.0 -22 0.0 3.281101

Aquesta és la informació que ens fa falta, quan llegeix la paraula Textura n'agafa el valor del costat, que és la ID de la textura i l'afegeix a la variable del registre:

```
sscanf(oneline, "Textura %d\n", &textu);
mon1.sector[loop_p].triangle[loop].tex=textu;
```

Ara ja només queda definir el registre vèrtexs amb les seves variables:

```
for (int vert = 0; vert < 3; vert++)
{
   readstr(filein,oneline);
   sscanf(oneline, "%f %f %f %f %f", &x, &y, &z, &u, &v);
   mon1.sector[loop_p].triangle[loop].vertex[vert].x = x/5;
   *tracta la y segons si tenim acivat/desactivat les parets*
   mon1.sector[loop_p].triangle[loop].vertex[vert].z = -(z/5);
   mon1.sector[loop_p].triangle[loop].vertex[vert].u = u;
   mon1.sector[loop_p].triangle[loop].vertex[vert].v = v;
}
```

Veiem com acaba els registres assignant els valors que va llegint al registre vèrtexs.
Recordem que tot això ho fa dins d'un bucle i que va omplint la informació des del primer triangle fins l'ultim.

La funció crea_mon():

Ara ja tenim tota la informació necessària carregada als registres, el que veurem que fa és agafar-la i dibuixar-la.
Per tal de veure-ho de forma més entenedora, anirem veient fragments de la funció i els anirem comentant.

```
void crea_mon (void)
{
  SetupWorld();
  numobjectes=mon1.numobjectes;
  for (int loop=0; loop<numobjectes+1; loop++)
  {
```

* Primer de tot fa el SetupWorld() per carregar la informació, i fem un bucle per anar recorrent tots registres de tipus mon1, que en el cas que estem comentant ara, només tenim l'habitació.

```
    numtriangles = mon1.sector[loop].numtriangles;
    for (int loop_m = 0; loop_m < numtriangles; loop_m++)
    {
```

* Ara fem un bucle per anar recorrent tots registres de cada triangle

```
        if (mon1.sector[loop].triangle[loop_m].tipus!=1
        {
```

*Si el triangle no forma part ni del Sostre ni del terra general...

```
            x_m = mon1.sector[loop].triangle[loop_m].vertex[0].x;
            y_m = mon1.sector[loop].triangle[loop_m].vertex[0].y;
            z_m = mon1.sector[loop].triangle[loop_m].vertex[0].z;
            u_m = mon1.sector[loop].triangle[loop_m].vertex[0].u;
            v_m = mon1.sector[loop].triangle[loop_m].vertex[0].v;
```

```
textu=mon1.sector[loop].triangle[loop_m].tex;
```

*Assignem les variables a la informació del registre vèrtex

```
glBindTexture(GL_TEXTURE_2D, textures[textu]);
```

* Prepara les textures per carregar-les

```
glBegin(GL_TRIANGLES);
glNormal3f( 0.0f, 0.0f, 0.0f);
```

* Aquesta és la instrucció per dir-li que dibuixi en forma de triangle

```
glTexCoord2f(u_m,v_m); glVertex3f(x_m,y_m,z_m);
x_m = mon1.sector[loop].triangle[loop_m].vertex[1].x;
y_m = mon1.sector[loop].triangle[loop_m].vertex[1].y;
z_m = mon1.sector[loop].triangle[loop_m].vertex[1].z;
u_m = mon1.sector[loop].triangle[loop_m].vertex[1].u;
v_m = mon1.sector[loop].triangle[loop_m].vertex[1].v;
```

* Li passem les variables que formen el vèrtex i les coordenades on ha de situar la textura

```
glTexCoord2f(u_m,v_m); glVertex3f(x_m,y_m,z_m);
x_m = mon1.sector[loop].triangle[loop_m].vertex[2].x;
y_m = mon1.sector[loop].triangle[loop_m].vertex[2].y;
z_m = mon1.sector[loop].triangle[loop_m].vertex[2].z;
u_m = mon1.sector[loop].triangle[loop_m].vertex[2].u;
v_m = mon1.sector[loop].triangle[loop_m].vertex[2].v;
glTexCoord2f(u_m,v_m); glVertex3f(x_m,y_m,z_m);
```

* Aquesta operació l'hem fet tres vegades, una per cada vèrtex que forma un triangle.

```
glEnd();
```

* Li indiquem que pari de dibuixar.

Ara ja visualitzarem l'habitació i podrem començar a editar-hi els mobles.
Cal recordar, que cada vegada que s'executa la funció display(), tot el procediment explicat es va repetint per tal d'anar mostrant constantment el dibuix per pantalla.

II.3.3.- Operacions sobre l'Habitació:

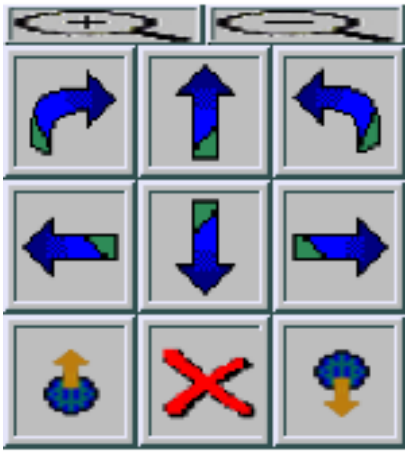
Una vegada carregat el dibuix de l'habitació en el Display Principal, ja podem treballar sobre ella sense calguer afer-hi cap moble, ja que tenim unes quantes opcions que afecten a tot el conjunt dibuixat, no només a un moble en concret.

L'habitació la podem moure, inclinar, rotar,... Per això farem servir el Tauler de Control.

Tauler de Control (sobre habitació):

Com hem explicat abans, el Tauler de Control és un conjunt de subfinestres, amb una textura associada, que treballen com si fossin botons.

Per a associar-hi una textura a cada subfinestra el que hem de fer és senzillament dibuixar un quadrat tant gran com la subfinestra i a aquest assorciar-hi la textura. El procés fa servir la funció "glBindTexture" però en aquest cas no agafa una taula amb variables com hem vist abans, sinó que agafa una textura fixa perquè ens interessa que la textura sigui sempre la mateixa.



Descriurem d'esquerra a dreta i de dalt a baix:

-Zoom +: Apropa l'Habitació.
-Zoom-: Allunya l'Habitació.
-Fletxa corva dreta: Rota l'Habitació amb sentit horari.
-Fletxa dalt: Habitació es mou cap a dalt.
-Fletxa corva esquerra: Rota l'Habitació amb sentit antihorari.
-Fletxa esquerra: Habitació es mou cap a l'esquerra.
-Fletxa baix: Habitació es mou cap a baix.
-Fletxa dreta: Habitació es mou cap a la dreta.
-Bola puja: Habitació rota sobre l'eix del horitzontal (de baix cap a dalt).
-Creu: Torna Habitació a punt de vista inicial.
-Bola baixa: Habitació rota sobre l'eix del horitzontal (de dalt cap a baix).

En la funció Display que està associada a la subfinestra Display Principal hi tenim:

```
glTranslatef(xp,yp,zp);
glRotatef(alpha, 1.0f, 0.0f, 0.0f);
glRotatef(beta, 0.0f, 1.0f, 0.0f);
```

Aquestes són les funcions OpenGL que fem servir per a moure l'habitació. A les 3 funcions els hi entren variables globals, de manera que si són canviades en altres punts del projecte modificaran la posició de l'habitació.

-El zoom funciona augmentant i disminuint la zp.
-El de esquerra/dreta augmentant i disminuint la xp.
-El de dalt/baix augmentant i disminuint la yp.
-El de puja augmenta l'alpha.
-El de baixa augmenta la beta.
-La creu inicialitza totes aquestes variables.

Exemple de callback (click de ratolí):

En aquest exemple la crida des de la definició de la subfinestra en la funció main seria "glutMouseFunc(mouse_zoom_in)". És a dir com a callback de ratolí interpreta la funció "mouse_zoom_in". En aquest cas és molt senzilla: en el cas de que es premi el botó esquerra del ratolí augmenta 0'75 a la variable global zp.

```
void
mouse_zoom_in(int button, int state, int x, int y)
{
	switch(button)
	{
		case GLUT_LEFT_BUTTON:
			if (state == GLUT_DOWN)
			{
				zp +=0.75f;
				redisplay_all();
			}
			break;
	}
}
```

- redisplay_all(): és una funció que fa una crida a totes les finestres dient que s'actualitzin totes. D'aquesta manera fem que totes les subfinestres puguin estar connectades.

Moviment amb Teclat Especial:

Cal comentar que tots els moviments que es poden fer amb el Tauler de Control també es poden fer amb el teclat, ja que la subfinestra Display Principal té un callback de teclat especial amb les següents opcions:

-Fletxa dalt: Habitació es mou cap a dalt.
-Fletxa baix: Habitació es mou cap a baix.
-Fletxa esquerra: Habitació es mou cap a l'esquerra.
-Fletxa dreta: Habitació es mou cap a la dreta.
-Tecla "a" o "A": Apropa l'Habitació.
-Tecla "z" o "Z": Allunya l'Habitació.
-Tecla "espai": Torna Habitació a punt de vista inicial.

Barra d'Eines:

La barra d'eines està formada per 7 subfinestres, de les quals 6 actuen com a botons i una està posada per pura estètica. Cada "botó" al ser clicat desplega un menú GLUT i dintre d'aquest menú hi podem tenir varies opcions.

Descripció dels botons:
· *Color de Fons:* Com el seu nom indica la seva funció és canviar el color de fons de la subfinesta on està dibuixada l'habitació. És útil quan ens trobem que el color de la textura de la paret o del terra és molt fosc i es confon amb el fons.

La funció que crea el color de fons del Display Principal és:

```
glClearColor(colors[iFondo][0], colors[iFondo][1], colors[iFondo][2], 1.0f);
```

Com es pot veure se li entren variables que es treuen de la taula "colors". Depenent del iFondo el fons serà d'un color o d'un altre. Aquesta taula "colors" defineix cadascun dels colors que es poden triar en el menú GLUT que surt al prémer el botó "Color de Fons", i estan definits amb RGB.

Els colors que es poden triar són:

- Negre.
- Verd fosc.
- Blau fosc.
- Blanc.
- Verd clar.
- Blau clar.

· *Afegir Moble:* Al prémer aquest botó s'obre un menú GLUT amb dues opcions:

- Mobles fet a mida: Si es tria aquesta opció s'executa el projecte "Mobles"[17], i és on es podrà triar i crear un moble fet a mida amb les variables que entri l'usuari.
- Mobles stàndar: Si es tria aquesta opció s'executa el projecte "Carregar_ASE"[18] i és on es podrà escollir entre una sèrie de mobles creats amb programes com el 3D Studio Max, Autocad,... i que tenen extensió .ASE.

· *Estat Habitació:* Si es prem aquest botó s'obre un menú GLUT amb dues opcions que fan referència a l'estat de l'habitació:

- Estat Final: És l'estat predeteriminat de l'habitació i és quan ens mostra l'habitació amb tots els mobles que li hem inclòs. Si mirem l'apartat de la part I "Dibuixar" que ens explica el fitxer de sortida, veurem que es creen 3 arxius de sortida, un sense sufix, l'altre amb sufix _FINAL i el tercer amb sufix _3D. En el cas de que no estiguem en aquest estat, per a arribar-hi hem de carregar l'arxiu .txt de sufix _FINAL.txt.
- Estat Inicial: És l'altre estat possible que pot tenir l'habitació, si s'escull aquesta opció en la subfinestra Display Principal ens mostrarà l'habitació sense cap moble, per aconseguir aquest estat el que fa l'aplicació és carregar un altre dels 3 arxius que crea el "Dibuixar", i és el que no té cap sufix.

Evidentment si no s'ha afegit cap moble la imatge resultant serà exactament la mateixa.

· *Vista:* Al prémer aquest botó ens apareix un menú GLUT amb 4 opcions, dues d'elles contenen un submenú GLUT. Les opcions tracten d'opcions referents tant al moviment com al resultat visual de l'habitació:

- Vista de la Planta: Aquesta opció és la predeterminada, i és la que ens mostra l'habitació vista des de la perspectiva de planta, és a dir vista des de dalt.

[17] II.3.4.- Mobles pàg. 79
[18] II.3.4.1.- Carregador_ASE pàg.79

Aquesta vista l'aconseguim en el moment de carregar l'habitació, igualant la variable de rotació "alpha" a 90. D'aquesta manera aconseguim una inclinació de 90º respecte el pla (x,z).

- Vista Interiror: És la vista des de dintre de l'habitació, ens serveix per a que l'usuari se'n faci una idea des d'un altre punt de vista, i a més també podem afegir els mobles o desplaçar-los. Aquest punt de vista l'aconseguim donant valor 0.0 tant a la variable "alpha" com a la variable "beta", que són les dues variables de rotació que tenim.
- Vista Lliure: Al situar-nos a sobre d'aquesta opció se'ns desplega un submenú amb dues opcions més:
    - Activar: Activa la variable "moviment", que fa que es pugui moure l'habitació al arrossegar el ratolí per sobre de la subfinestra Display Principal. Aquest moviment l'aconseguim al modificar la variable "alpha" amb el desplaçament vertical del ratolí per sobre de la subfinestra, i conjuntament al modificar també la variable "beta" amb el desplaçament horitzontal del ratolí per sobre de la subfinestra.
    - Desactivar: Desactiva la variable moviment. És l'estat que ve predefinit. Aquesta variable també es desactiva si volem afegir algun moble.

- Parets: També ens apareix un submenú amb les mateixes opcions:
    - Activar: Aquest és l'estat que ve predefinit. Al prémer activar, les parets, en el cas d'estar desactivades, agafen l'alçada que tenen realment. D'aquesta manera es té una visió més real de l'habitació. Cal comentar que el projecte "Dibuixar" guarda l'alçada de les parets a part perquè la variable Y (que seria la variable d'alçada) que guarda té sempre una alçada fixa de 0.05. Per tant si volem que es vegin les parets tal i com són hem de modificar aquesta variable Y que hi ha guardada en l'arxiu txt per l'alçada de la paret real, diferenciant el que són les portes i les finestres que ja tenen una alçada fixa. També s'activa automàticament al seleccionar la opció de "Vista Interior".
    - Desactivar: Si es vol desactivar la paret, el procés és simplement tornar a carregar l'habitació però amb les variables Y de cada punt originals, és a dir, amb valor de 0.05, d'aquesta manera tenim que les parets són molt baixes i només ens de referència, sense tapar-nos el contingut de l'habitació.

· *Ajuda:* Aquest botó ens executa l'ajuda, que es tracta d'un document html on s'explica a mode de manual d'usuari el contingut de cada part de programa.

· *Sortir:* Al acabar d'afegir tots els mobles es prem aquesta opció de la barra d'eines. Emmagatzema tota la informació de tots els mobles que s'hi ha afegit i ens torna a la interfície anterior. Aquest punt s'explicarà millor en el seu apartat[19].

Creació de menús i submenús GLUT:

Per a crear un menú i un submenú GLUT primer definirem el submenú i l'associarem a la funció que li dirà que ha de fer al ser seleccionat, en aquest cas onMenu5. Després l'omplirem amb els glutAddEntry, assignant un text i un tipus, que després cridarem amb un "swich".

```
menuLliure = glutCreateMenu(onMenu5);
glutAddMenuEntry("Activar", MOVIMENT);
glutAddMenuEntry("Desactivar", NO_MOV);
```

Després passarem a definir el menú, que també l'associarem a la mateixa funció que el submenú. L'omplirem amb les opcions que vulguem i li posarem en la posició que vulguem posar el submenú un glutAddSubMenu amb el text i el nom amb que l'hem definit. Finalment introduirem un glutAttachMenu(GLUT_LEFT_BUTTON) que serveix per a que al clicar el botó esquerra del mouse es desplegui el menú creat. També es pot posar GLUT_RIGHT_BUTTON si volguéssim que fos al clicar el botó dret,...

```
menuVista = glutCreateMenu(onMenu5);
glutAddMenuEntry("Vista de la planta", PLANTA);
glutAddMenuEntry("Vista Interior", TRES);
glutAddSubMenu("Vista Lliure", menuLliure);
glutAttachMenu(GLUT_LEFT_BUTTON);
```

[19] II.3.7.- El Fitxer de Sortida pàg. 105

II.3.4.- Mobles:

En aquest apartat explicarem els dos projectes destinats a carregar els mobles que es volen afegir a l'habitació. El projecte Carregar_ASE i el projecte Mobles.

La idea principal d'aquests projectes és que copiïn el moble seleccionat, ja sigui fet a mida (Mobles) o stàndar (Carregar_ASE), en un arxiu txt anomenat Objecte.txt, ja que aquest arxiu és el que tracta a tots els objectes que introduïm i editem a l'habitació, tal i com explicarem en apartats posteriors.[20]

II.3.4.1.- Carregador ASE:

Primer de tot cal fer l'apunt de que entenem com a moble ASE un moble stàndar.
La funció del Carregador ASE és preparar mobles que s'entren en el format d'arxiu .ASE. És un tipus d'arxiu que es permet exportar de programes de la talla del *3D Studio Max*, l'*Autocad*, ... això ens permet poder contar amb mobles d'un gran realisme.

El problema és que OpenGL no és capaç de traduir un arxiu amb aquest format, s'ha de crear un traductor que pugui interpretar un .ASE i el tradueixi a .txt amb l'estructura que nosaltres fem servir. Per tant ens veiem en l'obligació de crear una petita aplicació capaç de traduir aquest tipus d'arxius per a l'ús que en volem fer.

[20] II.3.5.- Afegir moble pàg 92.

De .ASE a .TXT:

L'estructura d'un arxiu ASE és:

- Enumeració i descripció cadascun dels materials i submaterials utilitzats amb tots els atributs possibles i molt detallats.
- Dades a tenir en compte, com el número de vèrtex (MESH_NUMVERTEX) i el número de cares (MESH_NUMFACES), que són els triangles per nosaltres.
- Llistat de vèrtex (MESH_VERTEX) enumerats i amb les seves coordenades:

```
*MESH_VERTEX  213 -52.9805      83.6450      75.6424
```

- Llistat de cares (MESH_FACE) enumerades, amb identificadors de vertexs (A, B, C), identificadors d'arestes (AB, BC, CA) , tipus de suavitzat (MESH_SMOOTHING) i el tipus de material que té la cara (MESH_MTLID).

```
*MESH_FACE  0:  A:  14 B:  13 C:  12 AB:  0 BC:  1 CA:  1
*MESH_SMOOTHING 17 *MESH_MTLID 0
```

Com podem veure hi ha bastant més informació que la que nosaltres necessitem. L'aplcació que hem fet llegeix fa les següents operacions:

- Llegeix els .bmp dels materials i submaterials utilitzats en l'ASE i els enumera tal i com els enumera l'ASE.
- Els localitzem en la Base de Dades i els hi associem la ID que té pel nostre programa.
- Llegeix i guarda el nombre de polígons que té per a crear un bucle.
- Creem un arxiu .txt amb el nom del moble, i hi escrivim:
    - "ASE": per a que es pugui identificar com a aquest tipus de moble quan el llegim.
    - L'identificador: és el mateix identificador que tindrà a la Base de Dades.
    - 10 camps per a textures: s'omplen amb l'identificador de cada textura que utilitza, si en sobre, que és lo normal, es posa "-1".

    - Nombre de polígons: s'hi escriu també el nombre de polígons que té el moble.
- Llegeix i interpreta cada cara una a una de la llista de cares de la següent manera:
    - Se li posa un comptador que servirà d'índex de cares.
    - Els vèrtexs (A,B,C) els va a buscar un a un a la llista de vèrtex per l'identificardor que hi té associat.
    - El primer vèrtex es pren com a origen i s'iguala a zero, a partir d'ara a cada vèrtex se li restaran les coordenades del primer vèrtex per assegurar-nos de que estigui centrat, ja que ens vam trobar que no totes les figures estan centrades en l'origen de coordenades.
    - El guardem en l'arxiu .txt amb l'estructura que nosaltres fem servir i que ens va millor per a que pugui interpretar OpenGL, però substituïm la coordenada Y per la coordenada Z ja que OpenGL té el pla (x,y) com pla horitzontal, no el (x,z) que tenen la majoria i per tant les figures quedarien girades 90º cap per avall. També associem als 3 vèrtexs la ID de la textura que ens ha donat el MESH_MTLID i que nosaltres hem interpretat.
    - Una vegada guardat el vèrtex tanquem ens situem l'inici de l'arxiu .ASE i anem passant línies fins a trobar la segona cara i repetim el procés fins que el comptador sigui igual al nombre de cares.
- Guarda les dades al .txt de sortida i ja estem preparats per a poder afegir l'arixu .ASE.

El Carregador ASE, en un principi era un projecte molt més transcendent del que ha acabat sent. Ja que abans se'n encarregava de tota la traducció i la interpretació de l'arxiu .ASE, i la creació del .txt. De manera que al seleccionar un moble de la llista de mobles ASE aquest llegia directament de l'arxiu ASE i el passava al txt que ens interessava.

La operació tenia el problema de que tardava moltíssim i després d'haver vist la funcionalitat de la Base de Dades, vam decidir que al donar d'alta un moble ASE ja fes la conversió de ASE a TXT ja que així només calia fer-la una vegada i d'aquesta manera el Carregador ASE només llegeix la ID del moble i en carrega l'arxiu TXT del moble en concret i va molt més ràpid.

La Interfície:

El seu nom en el projecte és Carregador_ASEDlg.
És on escollim quin moble stàndar col·loquem a l'editor d'habitacions.

Camps:
- DDX_Control(pDX, IDC_LIST1, Llista_ASE)
- DDX_LBString(pDX, IDC_LIST1, m_ASE)
- ON_BN_CLICKED(IDC_AFEGIR, OnAfegir)
- ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
- Botó Cancel·lar

Serveis:
- OnInitDialog: Carrega la llista amb els diferent tipus de mobles existents.

- OnSelchangeList1: Si seleccionem un tipus ens esborra la llista i la carrega amb els mobles del tipus escollit, si seleccionem un moble, ens n'agafa el nom per carregar-lo.

- OnButton1: Ens neteja el contingut de la llista i la torna a carregar amb el tipus de mobles.

- OnAfegir: Comprova que haguem escollit un moble i no un tipus o senzillament res, i si és correcte carrega el moble.

- B_Cancel·lar: Es tanca la interfície.

II.3.4.2.- Mobles a mida:

Aquest tipus de mobles són els que el client pot fer a la seva mida. Lo que seria el dibuix base d'aquests mobles els hem fet nosaltres construint-los o "dibuixant-los" en forma de malla 3D en un fitxer TXT. Cada moble té el seu propi TXT, es troben guardats dins la carpeta del projecte i els seus noms són cadira.txt, taula.txt i armari.txt. Tots ells els mostrem als annexes.

El principal problema que tenen aquests mobles és que al ser dibuixats per nosaltres no tenen el nivell de detall o realisme que tindran els mobles .ASE, creats a partir de programes especialitzats.

Vam fer aquests tres mobles ja que vam pensar que eren els que són més parametritzables, en el cas que no haguéssim aconseguit carregar els mobles ASE n'haguéssim intentat fer més per tal de poder omplir més les habitacions però per sort no ha estat el cas.

Passem a fer l'explicació separant-la per parts i finalment en comentarem les interfícies.

La Cadira:

El client pot fer-se a mesura una cadira partint d'una cadira base, on podrà assignar-hi les textures que hi trobi convenients.
Aquestes textures són dues, una seria la Estructura, que vindrien a ser les potes i les barres que aguanten el respaldo, i l'altra l'hem anomenat Respaldo, encara que a part del respaldo, també engloba el seient. Les escull en dues llistes diferents, i nosaltres li assignem una variable diferent a cada una.

Mostro un fragment mol petit del cadira.TXT, la primera línia de l'arxiu, un vèrtex del tipus Estructura i un vèrtex del tipus Respaldo per tal de comprendre com el modifiquem:

NUMPOLIGONS 40

Estructura 1
Textura 0
0.0 0.0 0.0 0.0 0.0
0.0 1.602 0.0 0.0 1.0
0.0712 0.0 0.0 1.0 0.0

Respaldo 19
Textura 1
0.0 0.801 0.0712 0.0 0.0
0.8722 0.801 0.0712 1.0 0.0
0.8722 0.801 0.78 1.0 1.0

El que fem és obrir dos arxius, el cadira.txt i l'objecte.txt, que és allà on passarem la informació modificada de la cadira i des d'on l'editor d'habitacions en carregarà la informació per tal d'afegir-hi el moble.

Anem a copiar d'un arxiu a un altre, llegint a un i escrivent a l'altre. Primer escrivim "CADIRA" a l'objecte.txt per tal de que l'editor d'habitacions reconegui que li entrarà una cadira, en copiem el NUMPOLIGONS 40, ja que el nombre de triangles no canviarà, però si que utilitzarem el 40 per fer un bucle i anar tractant cada triangle.

Com que l'editor d'habitacions ho requereix, anirem escrivent "Paret" un triangle si, un no.

Després del Paret, i abans de definir cada triangle hi hem d'escriure la ID de la textura, per això anem llegint l'arxiu i si llegeix "Estructura" hi escriu la ID de la textura associada a l'estructura, i si llegeix "Respaldo" hi escriu la ID de la textura que el client ha escollit per el respaldo.

Arribem al final del fitxer i ja tenim llesta a l'objecte.txt la cadira parametritzable.

La Taula:

Repetim un procés similar al de la cadira, del taula.txt copiarem a l'objecte.txt, aquest cop però la parametrització és diferent, ja que el client pot escollir més paràmetres.
Pot escollir dues textures, una per les potes i l'altre per lo que seria la base de la taula. A més també escull l'altura de les potes, la llargària i l'amplada de la base.

Per fer això, assignem la llargària a la variable x, l'alçada a la y i l'amplada a la z, ara anirem fent correspondre els punts (x, y, z) a aquestes variables.

Comencem escrivint "TAULA" a l'objecte.txt per tal de que l'editor d'habitacions reconegui que li entrarà una taula, en copiem el NUMPOLIGONS 44, i comencem el bucle per cada triangle.

Cal dir abans, que en el taula.txt, els vèrtexs no porten abans l'estructura de "Respaldo" o "Estructura" com hem vist amb la cadira sinó que es diferencien en els següents tipus: Pota1, Pota2, Pota3, Pota4, Superior, Inferior, LateralL i LateralA.

Mentre anem llegint el taula.txt, anem reconeixent de quin tipus de triangle es tracta, i a part d'escriure "Paret" abans d'escriure un triangle si i un no, tal i com hem fet amb la cadira, anirem tractant els vèrtexs depenent del tipus que siguin.

Si es tracta d'un triangle de tipus pota, primer de tot hi escriurem la ID de la textura que el client ha assignat a les potes i si és dels altres tipus, hi escriurem la ID de la textura que el client ha demanat per la base de la taula.

Anem a modificar els punts que formen la malla 3D, segons els tipus de triangles que anem llegint. Recordem que la taula està dibuixada per punts i que aquests punts estan dibuixats partint del (0,0,0), per tant hi hauran punts que modificarem aplicant les variables que ha demanat el client i hi hauran punts que quedaran iguals.
Qui parteix del punt (0,0,0) és la pota 1.

Anem a veure per cada tipus com assignem els punts o a 0 o a la variable del client, ja que després ho sumarem als punts del taula.txt

Pota1: x = 0
y = alçada
z = 0
Pota2: x = 0
y = alçada
z = amplada
Pota3: x = llargada
y = alçada
z = 0
Pota4: x = alçada
y = llargada
z = amplada
La resta de tipus són: x = llargada
y = alçada
z = amplada

Ara anem llegint els vèrtexs del triangle que acabem de llegir de la taula.txt i hi anem sumant les noves coordenades i escrivint a l'objecte.txt, a no ser que la coordenada original sigui 0, cosa que significa que és un punt a partir del qual estirem la taula, i si el modifiquéssim, la taula perdria la forma. En aquest cas escriure'm a l'objecte.txt la coordenada tal i com l'hem llegit a la taula.

Anem reccoren tots els triangles del fitxer repetint aquesta operació fins que arribem al final del fitxer i ja tenim llesta a l'objecte.txt la taula parametritzable.

L'Armari:

L'armari també es podria considerar una estanteria, el client li pot assignar la textura, que serà la mateixa per tot el moble, el nombre d'estants que tindrà, i que s'aniran repetint de forma vertical, i els centímetres cúbics que faran cada un dels estants on pot triar entre 25cm, 50cm o 1metre.

El procediment també consisteix en agafar la base que la tenim a l'armari.txt, aplicar-hi els canvis i copiant-ho a l'objecte.txt.

Primer de tot escrivim a l'objecte.txt la paraula "ARMARI" seguit per la ID de la textura, el nombre de repeticions i el tamany dels estants.
Després llegeix de l'armari.txt el nombre de polígons, els multipliquem per el nombre de repeticions, i ja tenim el nombre de polígons que formen l'armari sencer, ho escrivim a l'objecte.txt..

Per solucionar els centímetres cúbics el que hem fet, és que l'armari.txt està dibuixat com si l'armari fos d'un metre cúbic, per tant, si és de 50 centímetres en dividirem els punts entre dos, i si és de 25 centímetres, els dividirem entre 4 i si és d'un metre, els deixarem iguals.

Fem un bucle que s'anirà repetint tantes vegades com repeticions tingui l'armari, i a dins d'aquest, un altre bucle que serà el d'anar llegint els triangles i anar-los tractant i copiant un per un tal i com hem fet amb la cadira i la taula.
Recordant també el procés d'anar escrivent la paraula "Paret" un triangle si i un no.
Per tant, anirem llegint de l'armari.txt els triangles i els anirem copiant a l'objecte.txt, primer triangle a triangle, posant la textura triada per el client i fent les divisions corresponents amb els centímetres cúbics escollits, i després anirem repetint la operació tantes vegades com nombre d'estants tingui l'armari.

Un cop acabem el procés, ja tenim l'armari tot escrit a l'objecte.txt llest per ser executat.

Les Interfícies:

- Interfície Moble a mida:

El seu nom en el projecte és MoblesDlg.
És on veurem un llistat de mobles que es poden fer a mida.

Camps:
-DDX_Control(pDX, IDC_MOBLES1, m_mobles1)
-ON_LBN_SELCHANGE(IDC_MOBLES1, OnSelchangeMobles1)
-Botó Cancel·lar.

Serveis:
-OnInitDialog: Al obrir-se la finestra mostra una llista amb tots els mobles que es poden fer a mida.
-OnSelchangeMoble1: Al sel·leccionar un moble de la llista s'obrirà la finestra de creació del moble triat.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Cadira a mida:

El seu nom en el projecte és Cadira.
És on crearem la cadira a mida, les opcions són de triar 2 textures.

Camps:
-DDX_Control(pDX, IDC_CADIRA2, m_textura2)
-DDX_Control(pDX, IDC_CADIRA, m_textura1)
-ON_LBN_SELCHANGE(IDC_CADIRA, OnSelchangeCadira)
-ON_LBN_SELCHANGE(IDC_CADIRA2, OnSelchangeCadira2)
-ON_BN_CLICKED(IDC_previa, Onprevia)
-ON_BN_CLICKED(IDC_previa2, Onprevia2)
-Botó Acceptar.
-Botó Cancel·lar.

Serveis:

-OnInitDialog: Al obrir-se la finestra mostra 2 llistes amb els 2 tipus de textures que es poden triar, una pel respaldo i l'altra per l'estructura.

-OnSelchangeCadira: Al seleccionar una textura d'estructura de la llista, guardem el nom d'aquesta.

-OnSelchangeCadira2: Al seleccionar una textura de respaldo de la llista, guardem el nom d'aquesta.

-Onprevia: S'obre la finestra de Vista Prèvia on podem veure la textura que hem seleccionat de la llista de textures d'estructura.

-Onprevia2: S'obre la finestra de Vista Prèvia on podem veure la textura que hem seleccionat de la llista de textures de respaldo.

-OnOK: Si s'han triat les 2 textures corresponents creem la cadira. Primer llegeix de l'arxiu Cadira.txt, després modifiquem les textures i ho copiem a l'Objecte.txt, que és des d'on el tractarem.

-B_Cancel·lar: Es tanca la interfície.

- Interfície Taula a Mida:

El seu nom en el projecte és Taula.
És on escollim com volem que sigui una taula, feta a mida pel client.

Camps:
- DDX_Control(pDX, IDC_LIST1, m_tex1);
- DDX_Control(pDX, IDC_LIST2, m_tex2);
- DDX_LBString(pDX, IDC_LIST1, m_llista1);
- DDX_LBString(pDX, IDC_LIST2, m_llista2);
- DDX_Text(pDX, IDC_ALSADA, m_alsada);
- DDX_Text(pDX, IDC_AMPLADA, m_amplada);
- DDX_Text(pDX, IDC_LLARGADA, m_llargada);
- ON_BN_CLICKED(IDC_previa, Onprevia)
- ON_BN_CLICKED(IDC_previa2, Onprevia2)
- Botó Accepar
- Botó Cancel·lar

Serveis:
- OnInitDialog: Es connecta a la base de dades per tal d'omplir les llistes amb les seves textures corresponents, base taula i potes taula.
- OnSelchangeList1: Agafa el nom de la llista i el posa en un string, també marca que la llista ha estat seleccionada.
- OnSelchangeList2: Agafa el nom de la llista i el posa en un string, també marca que la llista ha estat seleccionada.
- OnOK: Comprova si hem especificat les mesures i si és correcte agafa l'arxiu taula.txt, hi aplica els canvis definits per l'usuari i copia la taula nova a l'arxiu objecte.txt.
También es connecta a la base de dades per trobar l'identificador de les textures seleccionades.
- Onprevia2: S'executa el projecte Vista Prèvia
- Onprevia: S'executa el projecte Vista Prèvia
- B_Cancel·lar: Es tanca la interfície.

- Interfície Armari a mida:

El seu nom en el projecte és Armari.
És on crearem l'armari a mida, les opcions són de triar la textura, el tamany (amb cm^3) i el número d'estanteries que es vol.

Camps:
-DDX_Control(pDX, IDC_LIST1, m_textura)
-DDX_Text(pDX, IDC_EDIT1, m_repet)
-DDX_Radio(pDX, IDC_TAMANY1, m_tamany)
-ON_LBN_SELCHANGE(IDC_LIST1, OnSelchangeList)
-ON_BN_CLICKED(IDC_previa, Onprevia)
-Botó Acceptar.
-Botó Cancel·lar.

Serveis:
-OnInitDialog: Al obrir-se la finestra mostra una llista amb les textures del tipus armari.
-OnSelchangeList: Al sel·leccionar una textura de la llista, guardem el nom d'aquesta.
-Onprevia: S'obre la finestra de Vista Prèvia on podem veure la textura que hem sel·leccionat de la llista de textures.
-OnOK: Si s'ha triat la textura, el número d'estanteries és més gran que 0 i s'ha triat un tamany creem l'armari. Primer llegeix de l'arxiu Armari.txt, després modifiquem les textures, les repeticions que hem de fer i el tamany i ho copiem a l'Objecte.txt, que és des d'on el tractarem.
-B_Cancel·lar: Es tanca la interfície.

II.3.5.- Afegir Moble:

Al afegir un moble ja sigui stàndar o fet a mida, venim dels carregadors Moble o Carregador_ASE respectivament, per tant ja tenim el moble amb la nostra estructura a l'arxiu Objecte.txt, així que ja podem treballar a partir d'aquí.

Situar el moble en l'espai:

Una vegada sortim de qualsevol dels carregadors, el següent pas és situar el ratolí a la part superior esquerra de la subfinestra Display Principal, clicar i mantenir clicat. Veurem que surt el moble que hem seleccionat transparent. Tot seguit l'anem arrossegant per la finestra i deixem anar el botó quan el tinguem situat al lloc desitjat. Aquesta acció la realitzem amb la següent línia de codi de la funció onMouseMotion:

```
veure_objecte2(x-x_0,y-y_0,375,1,0);
```

On se li entra:

- la posició de la x: que es troba de restar la posició de la coordenada x actual del ratolí de la posició de la coordenada x inicial del clic .
- la posició de la y: que es troba de restar la posició de la coordenada y actual del ratolí de la posició de la coordenada y inicial del clic.
- la posició d'alçada: que és 375, ja que respecte és l'alçada del nostre origen.
- una constant que explicarem més tard quan expliquem les glList.
- i la rotació: que sempre serà 0 perquè en el moment d'afegir no es pot rotar el moble.

Quan fem qualsevol acció sobre un moble, ja sigui afegir, moure, rotar,... sempre el copiarem a l'arxiu Objecte.txt per a tractar-lo a partir d'allà. Però es clar, un txt no és molt manejable així que necessitem guardar-lo en la memòria virtual per a poder-li modificar les variables,... Així que el que fem és copiar-lo dintre dels registres i més concretament en el Sector[0]. I això és el que fa la funció "SetupObjecte", llegeix de l'Objecte.txt, crea un nou món, el mon_objecte, que serà el món dels objectes i ho guarda a la posició 0 del Sector, és a dir, al primer Sector d'una manera molt semblant a la que ho fa el SetupWorld. La única diferència és que el SetupObjecte distingeix el tipus de moble que és, és a dir, si el moble és una Cadira a mida, un Taula a mida, un Armari a mida o un moble Stàndar. De manera que se li afegiran un tipus de variables o unes altres depenent del tipus de moble de que es tracti.

Un cop ja el tenim dintre del mon_objecte.sector[0] ja podem modificar-lo tant i com vulguem, ja que sinó ho féssim, tots els mobles es dibuixarien en l'extrem superior esquerra de la subfinestra Display Principal. I això és el que fa el "veure_objecte2", és a dir, modifica les coordenades inicials donant un desplaçament al moble que hem entrat.

Però aquí no acaba la funció del "veure_objecte2", sinó que també afegeix les dades del moble a un DiplayList.

Les DisplayLists:

Les DisplayLists són un conjunt de llistes que facilita l'OpenGL per tal de poder tractar amb objectes per separat. Cada objecte que s'afegeix en una DisplayList és una llista de dades que OpenGL pot interpretar. Amb la següent funció afegim un objecte al conjunt de llistes i li entrem també la posició que tindrà dins d'aquest conjunt i amb la de baix indiquem on s'acaba la informació que hem d'entrar:

```
glNewList(pos, GL_COMPILE);
glEndList();
```

Amb aquesta altra funció podrem cridar la llista on hi tenim l'objecte que volem que es mostri per pantalla:

```
glCallList(pos);
```

I amb aquesta altra ja podem esborrar la llista o les llistes on hi tenim l'objecte o els objectes que volem suprimir, ja que se li entra la posició de la llista i quantes més en volem esborrar (en aquest cas només esborraríem la llista situada a la posició pos) :

```
glDeleteLists(pos,1);
```

Com es pot veure el funcionament de les DisplayLists és bastant senzill i t'estalvia molts mals de cap.

Cal comentar que el veure_objecte2 es diu 2 perquè també hi ha el veure_objecte, que fa el mateix el que passa és que la primera mostra el moble transparent i el segon no. Això ho fa amb una funció que activa la transparència: "glEnable(GL_BLEND)" i una altra que la desactiva: glDisable(GL_BLEND). També es diferencien que el veure_objecte agafa la posició des d'una variable global mentre que la 2 li passen.

És important saber que mentre el moble està en la posició mon_objecte.sector[0] el moble és provisional. De manera que una vegada es deixi de clicar s'haurà de guardar en un altre lloc. També s'ha de dir que el mentre està en el mon_objecte.sector[0], el moble es guarda en la DisplayList(1) que és la primera, i també hi està de manera provisional.

Podem afegir un moble des de qualsevol perspectiva ja que sempre es desplaça pel pla (x,z), per tant per molt girada que estigui l'habitació el moble no ho notarà.

Afegir un moble a la Llista:

Al deixar de clicar confirmem que el moble està aproximadament allí on el volíem col·locar i passa el següent:

```
if ((state == GLUT_UP) && (afeg==true))
{
        crear_objecte(i,x,y,o);
        destruir_moble();
        afeg=false;
}
```

En aquest tros de funció veiem què passa quant deixem de clicar el botó esquerra del ratolí i a més a més estem afegint un moble. Primer es crea l'objecte, després destruïm l'objecte que estava guardat en la posició mon_object.sector[0] i deixem d'afegir.

La funció "crear_objecte" és molt important, ja que és la que ens permet distribuir la informació dels mobles que tractem.

El primer paràmetre del crear_objecte no influeix en aquest cas, ja que també el fem servir en altres llocs, la x i la y són les coordenades d'on hem deixat el moble i el 0 és l'alçada, que com hem dit, no podem aixecar el moble metre l'estem afegint.

L'aplicació té un comptador global anomenat "id_moble" que es va incrementant cada vegada que s'afegeix un moble, de manera que en tot moment sabem a quina posició afegir aquest nou moble i això ho aplica el crear_objecte, ja que la seva funció és primer que tot distingir quin tipus de moble és (Cadira, Taula, Armari o ASE) i després passar les dades del registre mon_objecte.sector[0] al mon_objecte.sector[id_moble] i això ens permetrà poder tornar a fer servir el mon_objecte.sector[0] per a tractar un altre moble.

Per a conèixer el tipus de moble i la posició que ocupa dintre de l'habitació utilitzem una taula anomenada ArrayMoble. L'omplirem de la següent manera:

- Si es tracta d'una Taula: afegirem un 1 en la posició id_moble que correspongui.
- Si es tracta d'una Cadira: afegirem un 2 en la posició id_moble que correspongui.
- Si es tracta d'un Armari: afegirem un 3 en la posició id_moble que correspongui.
- I si es tracta d'un ASE: afegirem u 4 en la posició id_moble que correspongui.

D'aquesta manera sabrem sempre quin tipus de moble tenim a cada posició.

El "destruir_moble" l'únic que fa és un glDeleteLists(1, 1). Així destruïm la primera llista del conjunt de llistes, que és on hi ha el moble provisional.

I per acabar deixem l'estat d'afegir.
D'altra banda, tenim la "Llista dels Mobles". Per explicar el seu funcionament cal explicar abans una funció que utilitzem molt sovint en tota la part d'OpenGL:

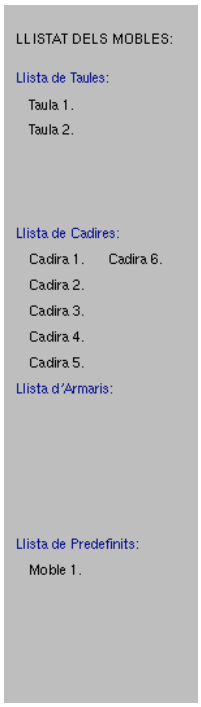
El redisplay_all i el glutPostRedisplay:

El glutPostRedisplay és una de les funcions més importants que incorpora la llibreria GLUT. Cada vegada que s'executa fa actualitzar la finestra que tenim activada en aquell moment. De manera que actualitza totes les variables que s'han modificat i redibuixa el contingut.

I el redisplay_all és una funció que fa tots els glutPostRediplay possibles, és una manera de comunicar-se amb totes les finestres a la vegada, d'aquesta manera si es fa alguna acció en una subfinestra GLUT i ens interessa que aquesta acció s'apliqui en 2 o 3 subfinestres, el redisplay_all serà el que els hi dirà que s'actualitzin:

```
void
redisplay_all(void)
{
   glutSetWindow(finestra);
   glutPostRedisplay();
   glutSetWindow(llista);
   glutPostRedisplay();
   glutSetWindow(planta);
   glutPostRedisplay();
   ...
}
```

Com veiem primer ens hem de situar en la finestra que volem fer el glutPostRedisplay gràcies a la funció glutSetWindow(nom de la finestra).



Pràcticament a cada callback que fem s'executa un "rediplay_all". La funció "onMouse" que és la que detecta les accions de clicar i deixar de clicar, en té un. I el Llistat del Mobles és una subfienstra, per tant s'actualitza al clicar i deixar de clicar.

El Llistat dels Mobles té associades unes funcions que recorren cada vegada la taula de mobles (ArrayMoble), així sap quins tipus de mobles hi ha i quants n'hi ha i els pot distribuir per l'espai de tota la subfinestra.

Com veiem distingeix el tipus de moble i depenent de la posició en que apareixen els va col·locant en el seu respectiu llistat. Aquest sistema ens anirà molt bé a l'hora de seleccionar, però ja ho veurem més endavant.

El Llistat dels Mobles l'hem dividit en cel·les per a poder inserir cada moble en una zona determinada. Cada cel·la té un identificador i un rang que ens permetrà saber on situar cada moble.

Escriure en una finestra GLUT

Una de les coses que no facilita no la llibreria GLUT ni OpenGL en general és poder introduir text en les seves aplicacions. La única manera de fer-ho no resulta gens fàcil, ja que s'han de crear unes funcions que puguin descriure la manera d'escriure un text en una finestra o en una subfinestra (com és el cas) GLUT.

Primer s'ha de descriure com són les fonts que utilitzarem, és a dir, tots els tipus de lletra i també el seu tamany i després adequar-lo a l'escala que fa servir GLUT per aquestes coses.

Seria aproximadament la mateixa idea que el LoadGLTextures, és a dir, a l'executar l'aplicació llegim i guardem tots els tipus diferents de fonts:

```
void setfont(char* name, int size)
{
   font_style = GLUT_BITMAP_HELVETICA_10;
   if (strcmp(name, "helvetica") == 0) {
      if (size == 10)
         font_style = GLUT_BITMAP_HELVETICA_12;
      else if (size == 16)
         font_style = GLUT_BITMAP_HELVETICA_18;
   } else if (strcmp(name, "times roman") == 0) {
      font_style = GLUT_BITMAP_TIMES_ROMAN_10;
      if (size == 20)
         font_style = GLUT_BITMAP_TIMES_ROMAN_10;
   } else if (strcmp(name, "8x13") == 0) {
      font_style = GLUT_BITMAP_8_BY_13;
   } else if (strcmp(name, "9x15") == 0) {
      font_style = GLUT_BITMAP_9_BY_15;
   }
}
```

I després hem de crear una altra funció a la que se li pugui entrar una posició de la subfinestra i una cadena de caràcters amb el text que es vol escriure:

```
void drawstr(GLuint x, GLuint y, char* format,...)
{
   va_list args;
   char buffer[255], *s;

   va_start(args, format);
   vsprintf(buffer, format, args);
   va_end(args);

   glRasterPos2i(x, y);
   for (s = buffer; *s; s++)
      glutBitmapCharacter(font_style, *s);

}
```

Gràcies a aquestes dues funcions ens resultarà molt més fàcil poder introduir textos en finestres o subfinestres.

Així que per a omplir el Llistat dels Mobles fem servir aquestes dues funcions. Però també per a escriure a cadascuna de les subfinestres que tenim en la barra d'eines.

II.3.6.- Operacions sobre el Moble:

Sobre el moble podem fer moltes operacions com desplaçar-lo horitzontalment, verticalment, elevar-lo, enfonsar-lo, rotar-lo cap a la dreta, rotar-lo cap a l'esquerra i fins i tot eliminar-lo.

Per a poder-li fer totes aquestes operacions però, primer necessitem seleccionar-lo ja que la idea és que l'habitació tingui més d'un moble, i la selecció es converteix en la manera de distingir sobre quin de tots el mobles volem operar.

Selecció:

Una vegada hem afegit un moble ja podem seleccionar-lo del Llistat dels Mobles. Per a fer-ho només cal clicar a sobre del moble. Com hem explicat abans el Llistat dels Mobles està distribuït en cel·les, on cada cel·la té un identificador i un rang, aquest rang ens serveix precisament per a que no ens calgui clicar exactament en un punt, sinó si es clica en qualsevol punt del rang el moble quedarà seleccionat.

Quan seleccionem el moble l'usuari veurà que el nom que se li dóna a la llista es posa vermell i també que un dels mobles que hi ha a l'habitació es posa transparent. A partir d'aquí el moble entrarà en l'estat de seleccionat.

Interiorment passen bastantes coses. Per començar hem de detectar a quina cel·la hem clicar, i d'aquí treure'n l'identificador. Tenint l'identificador sabrem quin tipus de moble i la posició en que es troba respecte el tipus al que pertany:

Els identificadors de les cel·les van del 1 al 40, distribuïts de la següent manera:

- 01-10: Taules.
- 11-20: Cadires.
- 21-30: Armaris.
- 31-40: Mobles Predefinits o ASE.

Com que tenim 10 mobles de cada tipus les desenes de l'identificador ens mostren el tipus de moble que és i les unitats ens diuen la posició dintre del tipus. Per tant al ser seleccionat un moble busquem a l'ArrayMoble tantes vegades com ens digui la unitat a la desena, sempre tenint en compte que un Taula és un 1, una Cadira un 2, un Armari un 3 i un ASE un 4.

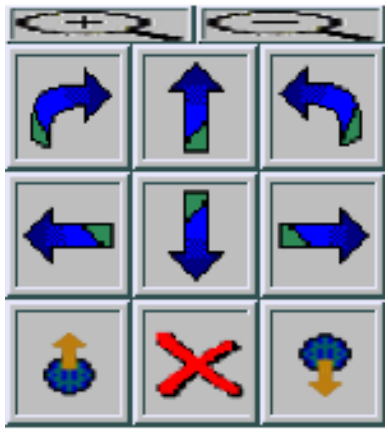
És a dir, si per exemple tenim l'identificador 24, vol dir que serà el quart armari que ens trobem. Per tant haurem de buscar a la taula ArrayMobles el quart 3 que ens trobem.

Un cop tinguem el moble localitzat hem de passar tota la informació d'aquest al mon_objecte.sector[0] i també a la glList(1). Així el podrem tractar, modicar-lo i veure'l transparent. Per aconseguir tot això he de copiar les dades del moble a l'Objecte.txt i a partir d'allà omplir el mon_objecte.sector[0] i la glList(1).

També hem de fer que el programa no dibuixi el moble que hi ha guardat a l'ArrayMoble, ja que sinó tindríem el moble transparent preparat per a ser modificat i la seva còpia en la posició que li correspon però sense ser transparent i no ens interessa. Per a fer-ho agafem i neguem el tipus, és a dir si el tipus és 3, passarà a ser -3, si és 2 a -2,... d'aquesta manera la funció "display" no el dibuixarà, sinó que se'l saltarà.

Quan ho tinguem tot copiat ja podrem començar a aplicar-li les operacions que es vulgui.

Totes les operacions que ara venen es duen a terme a partir del Tauler de Control:



Desplaçaments sobre el pla:

Per a fer el desplaçament horitzontal, sobre l'eix de les X, només cal comprovar primer de tot que el moble estigui seleccionat i si ho està:

- Si cliquem la fletxa de l'esquerra: El moble es desplaçarà cap a l'esquerra gràcies a la següent instrucció:

```
mon_objecte.sector[0].despx=mon_objecte.sector[0].despx-1.75;
```

  És a dir, li restem 1'75 píxels cada vegada que cliquem. Com veiem la modificació es guarda al mon_objecte.sector[0], i ho veurem per la subfinestra Display Principal ja que en la funció "display" llegeix i interpreta aquest desplaçament i canvia la matriu del model, així que aquest desplaçament només afectarà a l'objecte que tenim seleccionat.
- Si cliquem la fletxa de la dreta: El moble es desplaçarà cap a la dreta gràcies a la següent funció:

```
mon_objecte.sector[0].despx=mon_objecte.sector[0].despx+1.75;
```

  El funcionament és exactament que l'anterior, però aquest cop en lloc de restar 1'75 li sumem.

Per a fer el desplaçament vertical, sobre l'eix de les Y, només cal comprovar primer de tot que el moble estigui seleccionat i si ho està:

- Si cliquem la fletxa de dalt: El moble es desplaçarà de baix a dalt gràcies a la següent instrucció:

```
mon_objecte.sector[0].despx=mon_objecte.sector[0]. despy-1.75;
```

  És a dir, li restem 1'75 píxels cada vegada que cliquem. A part d'això, el funcionament no canvia.
- Si cliquem la fletxa de baix: El moble es desplaçarà de dalt a baix gràcies a la següent funció:

```
mon_objecte.sector[0].despx=mon_objecte.sector[0].despy+1.75;
```

  El funcionament és exactament que l'anterior, però aquest cop en lloc de restar 1'75 li sumem.

Rotacions:

Ara ja hem estat movent el moble per l'habitació una vegada ja estava afegit. Però no tots els mobles ens interessen amb la mateixa orientació, per això els podrem rotar cap als dos sentits:

- Si cliquem la fletxa de rotar a la dreta: El moble rotarà en sentit horari, i ho fa amb la següent instrucció:

```
mon_objecte.sector[0].rotar=mon_objecte.sector[0].rotar-15;
```

  Com veiem per a fer girar en sentit horari rotem el moble amb -15º, i el funcionament és igual que el desplaçament a l'esquerra. Amb la diferència que aquest cop tractem el paràmetre "rotar".
- Si cliquem la fletxa de rotar a l'esquerra: El moble rotarà en sentit antihorari i ho fa amb la següent instrucció:

```
mon_objecte.sector[0].rotar=mon_objecte.sector[0].rotar+15;
```

  És el mateix procediment, l'únic és que sumem en lloc de restar.

Elevacions:

El programa està obert a tot tipus de mobles, per tant per exemple si hi volem introduir un llum de sostre, una televisió, una estanteria,... necessitarem poder elevar el moble fins a certa alçada, ja que tots els moble venen predefinits en una alçada 0.

- Si cliquem al botó de la bola que puja: El moble s'elevarà, i ho fa amb la següent instrucció:

```
mon_objecte.sector[0].despv=mon_objecte.sector[0].despv+1.75;
```

  El funcionament és similar als anteriors, l'únic que en aquest cas modifiquem la variable que controla l'alçada i li sumem 1'75 píxels.

- Si cliquem al botó de la bola que baixa: El moble s'enfornsarà, i ho fa amb la següent instrucció:

```
mon_objecte.sector[0].despv=mon_objecte.sector[0].despv-1.75;
```

  Va com el pujar, però en aquest cas li restem 1'75.

Eliminar:

Finalment l'última operació que se li pot fer en un moble és la d'eliminar-se. De vegades s'afegeix un moble que no volem, o potser no l'hem parametritzat com esperàvem. La solució és eliminar-lo tant del Llistat dels Mobles, com de l'ArrayMoble, com de la glList i també del mon_objecte.sector[pos] on es trobi situat. Per a fer-ho utilitzem la funció "mouse_Borra".

El que fa la funció "mouse_Borra" és primer de tot mirar de quin tipus moble es tracta, i en quina posició està en l'ArrayMoble. Un cop localitzat, primer s'inicialitzen totes les variables possibles del mon_objecte.sector[0] i es fa un glDeleteLists, tant de la primera posició com de la posició del moble i en comptes d'esborrar-lo de l'ArrayMoble, com que afegim de manera seqüencial hi posem un "-200" en la seva posició, d'aquesta manera la funció "display" sabrà que no l'ha de mostrar a la subfinestra Display Principal.

Fent tots aquests passos l'usuari veurà que al clicar el botó X (creu) el moble seleccionat desapareixerà.

Desselecció:

Tant important és seleccionar com desseleccionar, ja que sinó només podríem treballar amb un sol moble i no és el que volem.

Hem de començar dient de que és l'estat per defecte, és a dir fins que no se selecciona un moble de la llista tots els mobles estan desseleccionats. Però l'aportació important d'aquest estat ve quan tenim un moble seleccionat, l'hem modificat (o no) i el volem desseleccionar, ja sigui perquè en volem seleccionar un altre o volem fer qualsevol altra acció.

Per a desseleccionar utilitzem l'antiga posició que tenia abans d'estar seleccionat (i com que l'estem tractant de manera provisional es troba en la primera posició tant de l'ArrayMoble, com de la glList, com de mon_objecte.sector), i li passem a la funció "Desseleccionar_Moble" juntament amb el tipus de moble que tractem. Aquesta funció se'n encarrega de buscar la seva antiga posició per l'ArrayMoble, una vegada trobada, fem el "crear_objecte" que ja hem explicat abans, passant-li aquesta posició de

l'ArrayMoble. El "crear_objecte" el que fa és copiar la informació del mon_objecte.sector[0] al mon_objecte.sector[el que li diguin] i també l'afegeix a la mateixa posició que li pertoca del conjunt de llistes (glList o DisplayLists).

D'aquesta manera tenim una altra vegada el moble en la seva posició inicial abans d'haver-lo seleccionat però amb les noves dades, a no ser que no s'hagin modificat. També cal mencionar que el moble al desseleccionar-se deixa d'estar transparent per a tornar a ser opac.

II.3.7.-El Fitxer de Sortida:

Anem a explicar ara com guardem la informació al fitxer de l'habitació del client un cop hi ha afegit mobles. En realitat, no només modificarem el fitxer de l'habitació del client (dni_nomhabitació_FINAL), sinó que al mateix temps també modificarem el fitxer que es llegeix a la Visita Interactiva[21] (dni_nomhabitació_3D), que té un tractament diferent.

Fins ara recordem que al fitxer hi tenim en forma de malla 3D, tota la informació de l'habitació, terra, parets, portes.....Tant al fitxer "_FINAL.TXT" com al "_3D.TXT".
Referint-nos al primer fitxer, hem de dir que ara quan guardem els mobles, no els guardarem en forma de malla 3D, amb els triangles i vèrtexs, sinó que n'escriurem la informació justa i necessària per tal de que quan reeditem l'habitació llegeixi aquesta informació i ens creï els mobles[22].
I per el segon, al tenir un tractament diferent, si que haurem d'anar-hi escrivent tota la informació dels mobles amb triangles i vèrtexs, definint-ne alguns paràmetres que explicarem a continuació.

Quan acabem d'editar l'habitació i sortim de l'aplicació farem aquest procés mitjançant una funció que vam anomenar Guardar_Mobles().

El primer que fa aquest fitxer és obrir un moment el fitxer "_final.txt" per tal de llegir-ne el nombre d'objectes, guardar el nombre en una variable i el tornem a tancar.
Després el tornem a obrir al mateix temps que el "_3D.txt", amb el mode d'escriptura i ens situem al final del text del fitxer "_3D.txt". Ara obrim un altre fitxer, el "dni_nomhabitació.txt", que és el que conté la informació sense cap moble, i en copiem la informació sencera al fitxer "_3D.txt."

El següent pas que farem serà recórrer l'ArrayMoble dins un bucle i anar escrivent als dos fitxers la informació necessària per a cada un d'ells. La recorrerem fins que no es trobi un 0 en una de les seves cel·les, que significarà que no hi hauran ja més mobles.
Si es troba un nombre més petit que el 0, també l'ignorarà ja que significa que en aquella posició hi havia un moble que després hem esborrat.

[21] II.4.- Visita Interactiva pàg. 111.
[22] II.3.8.-Reeditar habitació pàg. 109.

Anem recorrent la taula, si es troba un nombre superior a 0, ja n'agafem el nombre de registre que correspon a l'índex de l'ArrayMoble, en treiem la variable numtriangles, i al fitxer "_3D.txt" hi escrivim la paraula "Moble" seguit del nombre de triangles que acabem de llegir.

Ara segons el nombre superior a 0 que ens trobem dins la l'ArrayMoble, el tractarem segons li correspon, recordem que 1 significa taula, 2 és una cadira, 3 és l'armari i 4 és un moble de tipus ASE.

Anem a mostrar l'exemple d'una taula per fer-ho tot més entenedor:

```
if (ArrayMoble[loop_t]==1)
{
  tex1=mon_objecte.sector[loop_t].textura1;
  tex2=mon_objecte.sector[loop_t].textura2;
  px=mon_objecte.sector[loop_t].xpota;
  py=mon_objecte.sector[loop_t].ypota;
  pz=mon_objecte.sector[loop_t].zpota;
  desp_x=mon_objecte.sector[loop_t].despx;
  desp_y=mon_objecte.sector[loop_t].despy;
  desp_v=mon_objecte.sector[loop_t].despv;
  rot=mon_objecte.sector[loop_t].rotar;
  cad.Format("\nTAULA\n%i %i %f %f %f %f %f %f
%d",tex1,tex2,px,py,pz,desp_x,desp_y,desp_v,rot);
  Arxi.Write ( cad, cad.GetLength ());
```

*Fins aquí veiem com agafem la informació dels registres corresponents (el loop_t correspon amb l'índex de l'Array) i escrivim al fitxer "_FINAL.txt"(la seva variable en el codi és Arxi) la paraula "TAULA" i seguidament i en forma de línia, les variables per tal d'identificar com és la taula, que són, les dues textures, les variables per tal d'identificar la llargària, l'alçada i l'amplada de la taula, i seguidament el desplaçament per definir on es troba el moble situat dins l'habitació i el seu grau de rotació. Recordem que aquestes variables estan explicades a l'apartat crear objecte.

Veiem ara com tractem seguidament el fitxer "_3D.txt":

```
cad.Format("DESP %f %f %f\n",desp_x,desp_y,desp_v);
Arxi2.Write(cad,cad.GetLength());

cad.Format("ROTACIO %i\n",rot);
Arxi2.Write(cad,cad.GetLength());
desp_x=convertirX(desp_x);
desp_y=convertirX(desp_y);
desp_v=convertirX(desp_v);
```

*Escrivim la paraula "DESP" seguida per les variables de desplaçament, després la paraula "ROTACIÓ" seguit de la seva variable.

```
for (int loop_m = 0; loop_m < numtriangles; loop_m++)
{
tex1=mon_objecte.sector[loop_t].triangle[loop_m].tex;
if (par==false)
{
cad.Format("Paret %d \n",(loop_m+2)/2);
Arxi2.Write ( cad, cad.GetLength ());
par=true;
}
else
par=false;
cad.Format("Textura %d \n",tex1);
Arxi2.Write (cad, cad.GetLength ());
```

*Fins aquí escrivim la textura corresponent, i recordem també lo d'anar escrivint "Paret" un triangle si i un no.

```
for (int i=0; i<3; i++)
{
px = mon_objecte.sector[loop_t].triangle[loop_m].vertex[i].x;
py = mon_objecte.sector[loop_t].triangle[loop_m].vertex[i].y;
pz = mon_objecte.sector[loop_t].triangle[loop_m].vertex[i].z;
ux = mon_objecte.sector[loop_t].triangle[loop_m].vertex[i].u;
vx = mon_objecte.sector[loop_t].triangle[loop_m].vertex[i].v;
cad.Format("\n%f %f %f %f %f\n",px+desp_x,py+desp_v,pz+desp_y,ux,vx);
Arxi2.Write ( cad, cad.GetLength ());
}
```

*L'últim pas és escriure cada vèrtex amb les seves variables i aplicant-hi la suma de les variables de desplaçament, de forma que els punts es modifiquen i queden dibuixats allà on els hi toca dins l'habitació.
Recordem que estem dins un bucle i que anem fent aquests passos per a cada triangle.

El fitxer de sortida "_3D.txt" el donem per explicat, ja que tracta tots els mobles per igual, escrivent "Moble" sense fer distincions i dibuixant-ne la malla 3D corresponent.

Anem a veure doncs com el fitxer "_FINAL.txt" va tractant de manera diferent els mobles. Recordem que hem explicat fins que es troba un 1 dins l'ArrayMoble, que identifica com a taula, anem a veure com tracta la resta:

Si es troba un 2, significa que és una cadira, n'agafem les variables del registre i al fitxer hi escrivim la paraula "CADIRA", seguidament hi posem els ID's de les seves dues textures, les variables de desplaçament i la variable de rotació tal i com hem fet amb la taula

Si es troba un 3, l'interpretem com un armari, escrivim al fitxer "ARMARI" seguit de les seves pròpies variables, que són la textura, el nombre de repeticions de l'armari i la variable de divisió, que es refereix als centímetres cúbics. Després i tal com hem fet amb els mobles anteriors, escrivim les variables de desplaçament i la rotació.

Ens queda el moble ASE, que és el més senzill d'escriure, ja que no té variables parametritzables, quan la funció és troba un 4 a l'ArrayMoble, senzillament escriu al fitxer la paraula "ASE", seguidament l'identificador d'aquest moble a la base de dades i com sempre, les variables de desplaçament i rotació.

Ja tenim els fitxer preparats per quan necessitem reeditar l'habitació i fer la visita interactiva.

II.3.8.-Reeditar Habitació:

Ara explicarem com es carrega una habitació que no està buida, sinó que conté mobles.
El programa no només haurà de dibuixar l'habitació sinó que també haurà d'interpretar la informació dels mobles per tal de crear-los i situar-los en l'espai corresponent de l'habitació.

Un cop carregada l'habitació completa podrem aplicar-hi les operacions sobre l'habitació i sobre els mobles que ja hem explicat anteriorment. Per tant, en aquest apartat ens centrarem en explicar com interpreta la informació del fitxer per tal de mostrar l'habitació i els mobles.

Hem explicat ja com carreguem l'habitació sense mobles utilitzant les funcions SetupWorld i crea_mon. Recordem que la funció SetupWorld és qui carrega el fitxer i va omplint tots els registres, i després la funció crea_mon() és qui fa l'acció de dibuixar-los. Aquí la cosa canvia, ara serà la funció SetupWorld qui a part d'afegir tots els registres, dibuixarà els mobles, després la funció crea_mon, s'ocupa de dibuixar només l'habitació, per tant, qui treballa amb els mobles és la funció SetupWorld, reprenem doncs l'explicació d'aquesta funció.

L'habitació ara la carreguem igual, però al final ens trobem la informació dels mobles, dins del fitxer es pot trobar, segons el moble que toqui les paraules "CADIRA", "TAULA", "ARMARI" o "ASE".
Cada vegada que es troba una d'aquestes paraules, les tractarem de forma diferent, cada moble té una funció associada i que es diuen: Accio_Taula, Accio_cadira, Accio_Armari i Accio_ASE, posarem com a exemple què passa quan llegeix "TAULA" per tal de veure-ho tot de forma més entenedora:

```
else if ((sscanf(oneline, "TAULA\n")))
{
  readstr(filein,oneline);
  sscanf(oneline,"%d %d %f %f %f %f %f %f
%i",&tex1,&tex2,&pota_x,&pota_y,&pota_z,&desp_x,&desp_y,&desp_v,&rot);
  Accio_Taula(loop_p,tex1,tex2,pota_x,pota_y,pota_z);
  veure_objecte(desp_x,desp_y,desp_v,rot);
  crear_objecte(loop_p,desp_x,desp_y,desp_v);
}
```

Veiem com llegeix la paraula Taula, en llegeix les variables parametritzables i les de desplaçament i rotació, i crida la funció Accio_Taula passant-li totes aquestes variables. El que fa l'acció Taula és el mateix procediment que hem explicat anteriorment, resumint, agafa el fitxer taula.txt, en modifica els punts amb les variables que ha demanat el client i copia el moble final a l'objecte.txt.

Després veiem com és la funció SetupWorld qui ens dibuixa els mobles utilitzant les funcions veure_objecte i crear_objecte que ja em explicat anteriorment.

La cadira i l'armari el moble ASE funcionen exactament igual, només destacar que l'Acció_ASE fa una connexió a la base de dades per tal d'identificar amb la ID que llegeix al fitxer quin és el nom del fitxer que ha de carregar.

II.4.- El projecte Visita Interactiva:

La Visita Interactiva és un projecte anomenat Vista_3D. És on l'usuari podrà "passejar" per dintre de l'habitació que ha creat, ja sigui amb els mobles posats o sense. D'aquesta manera se'n podrà fer una idea molt realista de com li quedaria l'habitació amb els mobles nous.

Es tracta d'una finestra purament OpenGL, és a dir, sense utilitzar GLUT. De fet, com que no vam tenir constància d'aquesta llibreria que et simplifica bastant les coses fins a ben entrat el treball, la Visita Interactiva va ser la primera aplicació gràfica i l'hem deixat així per varies raons.

Una raó és que el sistema de programació és diferent, ja que al ser a més baix nivell, tens la opció de modificar molts més aspectes dels que pots fer-ho utilitzant el GLUT. Una altra raó és que hem comprovat que utilitzant només OpenGL, és a dir sense utilitzar GLUT, el programa va molt més fluït, i això suposem que es deu al fet de que no utilitza tantes rutines predeterminades, només utilitza les que realment necessita.

En el Visita Interactiva és l'únic lloc del projecte on es podrà veure el sostre amb la seva textura. I a més com que es pot sortir a l'exterior de l'habitació, per a que no sembli que estigui flotant hem incorporat un terra general.

Crear la finestra OpenGL:

Com hem dit, sense fer servir el GLUT, el carregar l'escena a la finestra és més complicat que posar un glCreateWindow com fem amb el GLUT, s'han de definir tots els atributs de la finestra:

```
hInstance = GetModuleHandle(NULL);
wc.style = CS_HREDRAW | CS_VREDRAW | CS_OWNDC;
wc.lpfnWndProc = (WNDPROC) WndProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hInstance;
wc.hIcon = LoadIcon(NULL, IDI_WINLOGO);
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground= NULL;
wc.lpszMenuName = NULL;
wc.lpszClassName = "OpenGL";
```

Com es pot veure hi ha forces variables per a definir la finestra, des del style, al tipus de punter, el tipus de fons,... La majoria d'elles les anul·lem perquè no ens interessa fer-les servir.

També s'ha de descriure com es vol la formació dels píxels, tots els missatges d'error possibles, cosa que en el GLUT ja ho incorporen les rutines predefinides. I fins i tot com tancar la finestra. També hem de descriure tots els tipus de callbacks que necessitem, com obrir/tancar la finestra, prémer una tecla, deixar-la de prémer i redimensionar la finestra.

Perquè cal dir que en aquest projecte comptem amb la opció de redimensionar la finestra i fins i tot passar-la a pantalla completa, ja que al executar l'aplicació Visita Interactiva, la finestra surt relativament petita. Es pot doncs, maximitzar o bé prement la tecla F1 passar a pantalla completa. Això ho aconseguim gràcies a la funció ReSizeGLScene, que té la rutina predefinida de glViewport, a la que se li entren les variables del tamany que volem la finestra, i se'n encarrega de tranpolar els píxels inicials.

Carregar l'Escena:

Per a carregar l'escena cal dir que obrim el fitxer amb el sufix _3D, ja que al no utilitzar el GLUT, no podem fer servir les DisplayLists i per tant necessitem tots els punts de les figures que la content. I el fitxer _3D surten descrits tots els mobles i tota l'habitació punt a punt. També surt el grau de rotació, ja que el punts els guardem sense estar rotats, per a que roti només l'objecte i no tota l'escena utilitzem el glPushMatrix cada vegada que incorporem un moble a l'escena, d'aquesta manera cada moble té la seva pròpia matriu de rotació.

Per a carregar tota l'habitació es fa servir una funció pràcticament igual que la que fem servir en l'Editor d'Habitacions, i l'anomenem "SetupWorld". La diferència és que aquesta llegeix primer tota l'habitació i quan arriba als mobles segueix llegint punt a punt. És a dir tenim un sol món, el mon1. Recordem que en l'Editor d'Habitacions teníem un món per l'habitació i un món pels objectes.

Moviment per l'Habitació:

Una vegada creada la finestra i carregada l'escena ja podem començar a moure'ns per l'habitació. Els moviments són una mica diferents que els de l'Editor d'Habitacions, i ens movem únicament amb el teclat.

Per a desplaçar-nos per l'habitació només ho podem fer amb el cursor i amb les fletxes "amunt" i "avall", els altres moviments no ens permeten desplaçar-nos perquè són des d'un punt fix. Per entendre bé el moviment hem de tenir clar que és com si estiguéssim dintre de l'habitació i volguéssim caminar per dintre, és a dir, nosaltres som el centre:

- Fletxa amunt: Avancem per l'habitació. Per fer-ho més realista hem incorporat la funció "boia" que sembla que caminem i a més per a no perdre les proporcions a mesura que avancem pel pla (x,z) necessitem fer funcions trigonomètriques:

```
xpos -= (float)sin(heading*piover180) * 0.05f;
zpos -= (float)cos(heading*piover180) * 0.05f;
if (walkbiasangle >= 359.0f)

{
```

```
        walkbiasangle = 0.0f;

    }
    else
    {
        walkbiasangle+= 10;
    }
    walkbias = (float)sin(walkbiasangle * piover180)/20.0f;
```

Per molt que sembli que som nosaltres els qui ens movem, en realitat el que es mou és l'habitació, i al avançar per dins d'ella necessitem calcular el sinus de la x i el cosinus de la z, ja que és pel pla on ens movem, d'aquesta manera sempre trobarem el mateix angle, la mateixa relació d'aspecte. Aquest càlculs es combinen amb el heading que vindria a ser l'alçada a la que ens trobem i el piover180 que és una constant que hem definit abans:

piover180 = 0.0174532925

El walkbiasangle ens serveix per ajudar al walkbias, que és el que fa la funció "boia", que per a que no es lïi aquest últim necessitem saber quan fem la volta completa sobre nosaltres mateixos, sinó l'efecte boia no funcionaria bé.

- Fletxa avall: Retrocedim per l'habitació. El funcionament és igual que l'anterior, però evidentment a la inversa.

```
    xpos += (float)sin(heading*piover180) * 0.05f;

    zpos += (float)cos(heading*piover180) * 0.05f;

    if (walkbiasangle <= 1.0f)
    {
        walkbiasangle = 359.0f;
    }
    else
    {
        walkbiasangle-= 10;
    }
    walkbias = (float)sin(walkbiasangle * piover180)/20.0f;
```

Com veiem el codi és pràcticament igual que l'anterior, amb la diferència que aquí sumem en lloc de restar.

- Fletxa esquerra: És com si giréssim el cap cap a l'esquerra, és a dir, no ens movem però l'habitació rota en sentit antihorari al nostre voltant. Per a fer-ho tenim una variable de rotació sobre la Y (heading) que l'augmentem mentre apretem el botó.
- Fletxa dreta: És com si giréssim el cap cap a la dreta, és a dir, no ens movem però l'habitació rota en sentit horari al nostre voltant. Per a fer-ho el mateix que l'anterior per a la inversa, és a dir, disminuïm el "heading".
- Av.Pàg: És com si miréssim cap avall. Si el mantenim apretat l'habitació rotarà sobre l'eix horitzontal al nostre voltant. Augmentem la variable de rotació sobre la X (lookupdown) i l'augmentem.
- Re. Pàg: És com si miréssim cap a munt. Si el mantenim apretat l'habitació rotarà soblre l'eix horitzontal, però en sentit invers a l'anterior, al nostre voltant. Per a fer-ho fem el mateix que l'anterior, però disminuint.

II.5.- El projecte Vistes:

L'objectiu d'aquesta part del programa final, era mostrar una finestra on poguéssim veure diferents perspectives de l'habitació.
La millor manera de veure'n el resultat final és fent la visita interactiva però vam pensar que no estaria de més tenir també una finestra on poguéssim veure amb un sol cop d'ull com queda l'habitació de forma general.

La finestra és de tipus GLUT i inclou quatre subfinestres.
Per tal d'identificar quina és l'habitació que ha de mostrar quan rep l'ordre d'execució, ha de seguir el procediment d'obrir el fitxer PLANTA_ACTUAL.txt, on hi troba el nom de l'habitació que ha de mostrar.

El que fa cada subfinestra és carregar l'habitació amb els seus mobles corresponents tal i com hem explicat anteriorment amb l'editor d'habitacions[23], i per cada subfinestra canvia el punt de vista, oferint-ne per cada una, una visió diferent de l'habitació.
Cal a dir que ens carrega l'habitació amb les parets però sense el sostre, ja que aleshores no podríem veure'n les perspectives des de dalt.

Si les vistes que es generen de forma automàtica no ens convencen o senzillament les volem millorar, podem realitzar operacions sobre cada una de les subfinestres per tal de millorar-ne la perspectiva.

Per fer aquestes operacions, hem de situar el ratolí sobre la subfinestra que volem modificar i escollir entre les següents opcions:
Tecla "a": Pugem tota l'habitació cap amunt.
Tecla "z": Baixem tota l'habitació cap avall.
Cursor "esquerra": Movem tota l'habitació cap a l'esquerra.
Cursor "dreta": Movem tota l'habitació cap a la dreta.
Cursor "amunt": Desplacem l'habitació cap al fons. (Zoom allunyar).
Cursor "avall": Apropa l'habitació. (Zoom acostar).

Per aconseguir fer aquest projecte cada subfinestra opera de forma independent a les altres, cada una té la seva pròpia funció display, amb una ordre de perspectiva, Translate i Rotate diferents. Quan movem les diferents vistes el que fem és canviar les variables del Translate per tal de moure'n l'habitació en la subfinestra que tinguem com a activa en aquell moment.

[23] II.3.8.- Reeditar habitació pàg. 109.

II.6.- El projecte Vista Prèvia:

És el projecte més senzill de tot el programa.
Es tracta d'un reforç per tal d'ajudar l'usuari a escollir una textura. A cada interfície on apareix un llistat de textures, hi trobem a sota un botó que ens executa aquest projecte, i abans de l'ordre d'execució (ShellExecute), obre el fitxer nom_textura.txt i hi escriu el nom de la textura escollida. D'aquesta manera, el primer que fa el projecte un cop executat és obrir-ne el fitxer i amb l'ajuda de la base de dades saber quina és la textura que ha de mostrar.

El que ens apareix és una finestra petita de tipus GLUT, on es mostra un cub recobert amb la textura que ha de mostrar, d'aquesta manera l'usuari veu com és la textura per tal de tenir-ne una idea abans de fixar-la.

Per tal de dibuixar el cub, utilitzem la ordre glBegin(GL_QUADS). L'OpenGL identifica que volem dibuixar quadrats, i en definim els vèrtexs en forma de cub i amb les coordenades de textura correctament definides.

- PART III -
LA BASE DE DADES

III.1.-Explicació General

-Introducció

Quan vam tenir la idea de fer aquest programa, en un principi només havíem pensat en el que era l'editor d'interiors. Havíem d'aprendre primer de tot com funcionava l'OpenGL i el Visual C++. No va ser fins al cap d'un temps que vam descobrir la necessitat d'incorporar una base de dades.

Va començar quan vam crear els nostres primers móns amb OpenGL , se'ns va plantejar un problema. Un cop acabada la creació i el disseny de l'interior, què en fèiem?.

Una opció era plantejar-se el programa com una opció purament visual, on poguéssim crear l'espai i posteriorment tornar-lo a obrir per reeditar-lo. Es podria guardar l'arxiu amb un nom i després per reeditar, obrir-lo amb un carregador.

La idea però, va ser ampliar aquest concepte a una visió més comercial, així donàvem al projecte molt més sentit i utilitat.

Amb la base de dades, podríem portar una gestió de clients i que cadascun tingués i guardés les seves pròpies habitacions. També passava a ser un programa orientat a una empresa.

El problema era que la part d'OpenGL portava feina i era lo més important de tot el programa, per tant, vam decidir aparcar la base de dades quan feia aquestes mínimes funcions, a l'espera de saber si hi podríem dedicar més temps o no.

Al final, la base de dades ha canviat totalment, no només ella mateixa, sinó que també ho ha fet amb el concepte general del programa. Ara porta tota la gestió de clients i de les habitacions, registra i calcula les factures i realitza un parell de consultes molt útils de cara a l'empresa que utilitza el programa.

Cal a dir que no és una base de dades que s'ocupi d'una gestió al cent per cent professional o que sigui equiparable a un projecte dedicat únicament a una gestió similar. Per arribar a aquest nivell s'hi haurien de fer afegits, però val a dir que tenint en compte que ni tant sols contava en la idea inicial del projecte, fa que tot quedi molt ben lligat, i compleix moltes més funcions que les esperades en un principi.
A part de lo comentat, la base de dades també va facilitar un canvi bastant important en el programa, que és la inclusió de poder afegir textures i mobles (ASE). La idea era que

el programa es presentava amb una àmplia llibreria de textures i mobles tancada però vam fer un afegit important canviant bona part del codi i amb l'ajut de la base de dades. Ara es poden afegir les textures i els mobles que el propietari del programa desitgi.

Afegint textures s'amplien molt més les possibilitats visuals i la realitat de cada habitació, per exemple, un client pot portar fotografiades com són les seves parets, terra o portes i la seva habitació quedarà molt més realista que agafant una textura similar dins una llibreria tancada.
Si podem també entrar mobles (ASE), permet que si l'empresa afegeix nous mobles en les seves línies de venda, el programa no quedi antiquat, és a dir, que es tinguin mobles a la venda que el programa no pot mostrar gràficament perquè té una llibreria limitada de mobles.

- Part Tècnica

Les dades s'emmagatzemen utilitzant el *Microsoft Access*. Les diferents taules que hi tenim, no estan dissenyades Orientada a Objectes, només són taules carregades d'informació, sense relacions ni restriccions entre elles. Però amb el Visual C++, si que hem definit com una mena d'objectes, que en realitat són classes genèriques ique es refereixen cada un d'ells a una de les taules de l'Access. Aquests són: l'O(objecte)Client, l'OMoble, l'OPlanta i l'OTextura.
Contenen totes les funcions associades a les taules de la base de dades, i aquests si que tenen restriccions entre ells, per exemple, si es dona de baixa un client, també s'esborren totes les seves habitacions, a no ser que tingui alguna factura pendent.

S'utilitzen també una sèrie d'interfícies que són de la classe Form View. Són qui interaccionen amb l'usuari i són qui utilitzen les funcions que tenen els objectes.

Finalment, per fer la interacció entre l'aplicació Visual C++ i l'emmagatzament de dades amb el *Microsoft Access* utilitzem el mecanisme ADO (ActiveX Data Objects) , que també era nou per a nosaltres i que expliquem en l'apartat següent.

Cal a dir també que hi ha altres projectes dins el programa que també es connecten a les dades emmagatzemades, per fer-ho utilitzen també l'ADO i en veurem la relació en el diagrama de classes que conté aquesta primera part de la memòria.

Durant la memòria d'aquesta primera part anirem explicant tot l'anàlisi i el disseny de la base de dades.

III.2.-ADO (ActiveX Data Objects)

L'ADO es un mecanisme utilitzat pels programes per comunicar-se amb les bases de dades, donar-li ordres i obtenir resultats d'elles.

Permet manipular la pròpia base de dades per crear noves àreas per l'emmagatzament d'informació (taules), com també alterar o eliminar les ja existents, entre altres coses.

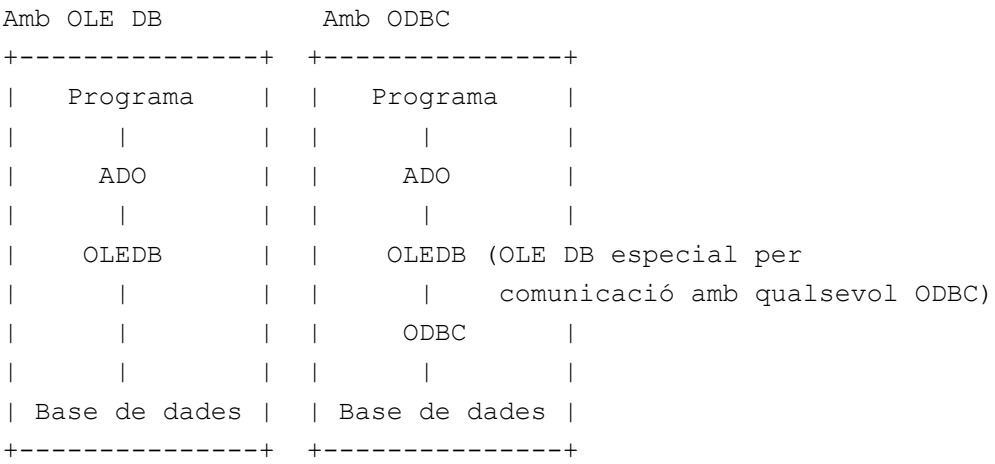
Va ser desenvolupat per Microsoft i és utilitzat en entorns Windows per llenguatges de programació com ara el nostre: Visual C++, com també es pot utilitzar en la Web.

ADO es comunica amb la base de dades a través d'un "proveïdor de dades" i la base també respon a través d'aquest proveïdor:

Programa ---> ADO ---> Proveïdor de dades ---> Base de Dades

Programa <--- ADO <--- Proveïdor de dades <--- Base de Dades

La connexió ADO pot utilitzar dos tipus de proveïdors de dades, OLE DB i ODBC. També cal dir que cada tipus de base de dades (Access en el nostre cas) té el seu propi proveïdor específic de tipus OLE DB i que en el nostre cas es diu "Microsoft.Jet.OLEDB.4.0.". Veiem ara gràficament la diferència entre un proveïdor OLE DB i ODBC:



Tot això és transparent a l'usuari d'ADO, que no té perquè adonar-se ni conèixer aquests mecanismes.

Components d'ADO que utilitzem:

- **Connection:** (Permet establir una connexió amb la base de dades)

  La **connexió** es com una autopista que permet el flux de dades entre el programa i la base de dades. Per ella poden viatjar les ordres que des del programa s'utilitzen per fer sol·licituds d'informació a la base de dades o per realitzar una operació dins d'ella com obrir i tancar la base de dades, esborrar registres, afegir registres, modificar taules, etc..

- **El RecordSet:** (Manipula un conjunt de registres de la base de dades)

El Recordset és, com el seu nom indica, un conjunt de registres. En general, les seves dades tenen el seu origen en una base de dades, encara que també poden generar-se independentment d'aquesta.

Un recordset pot contenir zero o més registres. Cada recordset té una col·lecció de camps, que es comú a tots els records. Podem veure'l com una matriu o taula, on les files son els records, i les columnes son los camps.

```
Exemple de Recordset amb dades d'una taula:

+------------+---------+----------+
| IdClient   | Nom     | Cognom   |
+------------+---------+----------+
|     1      | Lluis   | Pérez    |  <-- Record 1
+------------+---------+----------+
|     5      | Josep   | Abreu    |  <-- Record 2
+------------+---------+----------+
|     3      | Pere    | León     |  <-- Record 3
+------------+---------+----------+
|     7      | Maria   | Marcano  |  <-- Record 4
+------------+---------+----------+
      |          |          |
      |          |          +------- Camp "Cognom"
      |          |
      |          +------------------ Camp "Nom"
      |
      +---------------------------- Camp "IdClient"
```

El recordset, té capacitats de navegació entre el seu conjunt de registres. Pot:

- Moure's al següent registre

- Moure's al anterior

- Moure's al primer

- Moure's a l'últim

- i altres.

En un recordset, es veu i es poden editar les dades d'un sol registre en un temps donat, es poden manipular les dades dels camps del "registre actual" on es troben.

A més d'editar registres, també es pot:

- Afegiregistres nous

- Esborrar registres

L'edició, la inserció i l'esborrat de registres en el recordset, es reflectiran en la Base de Dades.

Altres funcions importants del RecordSet:

GetFieldValue: Agafa el valor d'un element del registre concret.

SetFieldValue: Modifica o afegeix l'element d'un registre concret.

AddNew: Per afegir un registre.

Delete: Per esborrar un registre.

Edit: Per modificar un registre.

MoveNext: Per moure's al següent registre.

IsEof: Estem al final de la taula?.

III.3.-Especificació de requeriments

La Base de Dades s'ocupa de dur a terme la gestió d'habitacions que dissenya l'empresa, així com també s'ocupa de gestionar els clients, els mobles pertanyents, i també les textures que formen part del programa.
Abans de poder gestionar les habitacions, necessitarem primer tenir els clients, els mobles i les textures, ja que són els qui formen les habitacions.
Per tant, en el moment d'inicialitzar per primer cop el programa i la base de dades el millor seria primer entrar els mobles i les textures per tal de poder dissenyar bé les habitacions i després els clients que són a qui pertanyen aquestes habitacions.

Els **Mobles** es donen d'alta i es poden llistar i per consultar-ne la informació de tots, l'stock que en tenim a l'empresa i el preu. Un moble pot tenir varis tipus depenent de a quin o quins tipus d'habitació pertany. Els Mobles s'identifiquen per un ID que li assigna automàticament la base de dades.

Les **Textures** s'han de donar d'alta, indicant-ne el nom i a quin tipus de superfície pertany. També es poden llistar per tal de veure'n tota la informació. Les Textures s'identifiquen per un ID que li assigna automàticament la base de dades.

Els **Clients** s'han de donar d'alta, indicant-ne la seva informació. També es poden donar de baixa o modificar-ne les dades (no pot canviar el seu DNI) i es poden llistar per tal de veure'n tota la informació. Els clients s'identificaran per el DNI.

Les **Habitacions** s'han de donar d'alta, indicant a quin client pertany i quin nom li posem. Un cop indicada la informació, passem a fer-ne el dibuix arquitectònic i a editar-hi els mobles. Després de guardar-se, ens interessa saber el preu total dels mobles i si la factura ha quedat pagada o no.
També podem donar de baixa l'habitació, i llistar-les per veure la informació de totes elles.
Una habitació s'identifica per la seva referència, que assigna automàticament la base de dades utilitzant el DNI del client i el nom de l'habitació.

III.3.1.-Informació que ens interessa guardar:

Dels **Mobles** ens interessa guardar-ne el nom, l'stock actual i el seu preu. També de quin tipus és, que es pot classificar en els següents: Cadira, Taula, Sofà, Llum, Llit, Armari, Cuina, Lavabo i Altres. Si un moble pot pertànyer a més d'un grup, es marca més d'un tipus. A part, la part gestora s'ocupa d'assignar-li un identificador. També s'ocupa de convertir el format ASE al format del nostre programa i de guardar-lo a la carpeta corresponent per al seu posterior ús.
També hem de poder-ne consultar l'stock per tal que no se'ns acabin.

De les **Textures** ens interessa guardar-ne el nom i el nom . També de quin tipus de superfície és, que es pot classificar en els següents: Cadira: pot ser de tipus Respaldo o Potes o les dues, de tipus Taula: pot ser de tipus Potes o Base o les dues, Terra, Paret, Sostre, Porta, Finestra, Armari o per als mobles ASE. Si una textura pot pertànyer a més d'una superfície, es marca més d'un tipus. A part, la part gestora s'ocupa d'assignar-li un identificador. També s'ocupa d'agafar l'arxiu .BMP i de guardar-lo a la carpeta corresponent per al seu posterior ús.

Dels **Clients** ens interessa guardar el DNI, el nom complet, el telèfon, l'adreça, i la data en la qual es va donar d'alta.

De les **Habitacions** ens interessa guardar el seu nom, a quin client pertany, la data en la que va ser creada, el preu total que costa i si la factura ha estat pagada.
La base de dades li assigna automàticament la referència agafant el nom de l'habitació i el dni del client.
Un cop creada l'Habitació hem de poder obrir-la per tal de reeditar-ne els mobles, generar la factura i veure'n la visita interactiva i les diferents vistes.
També hem de poder consultar quines estan pendents de pagament i modificar-ho quan el client pagui.

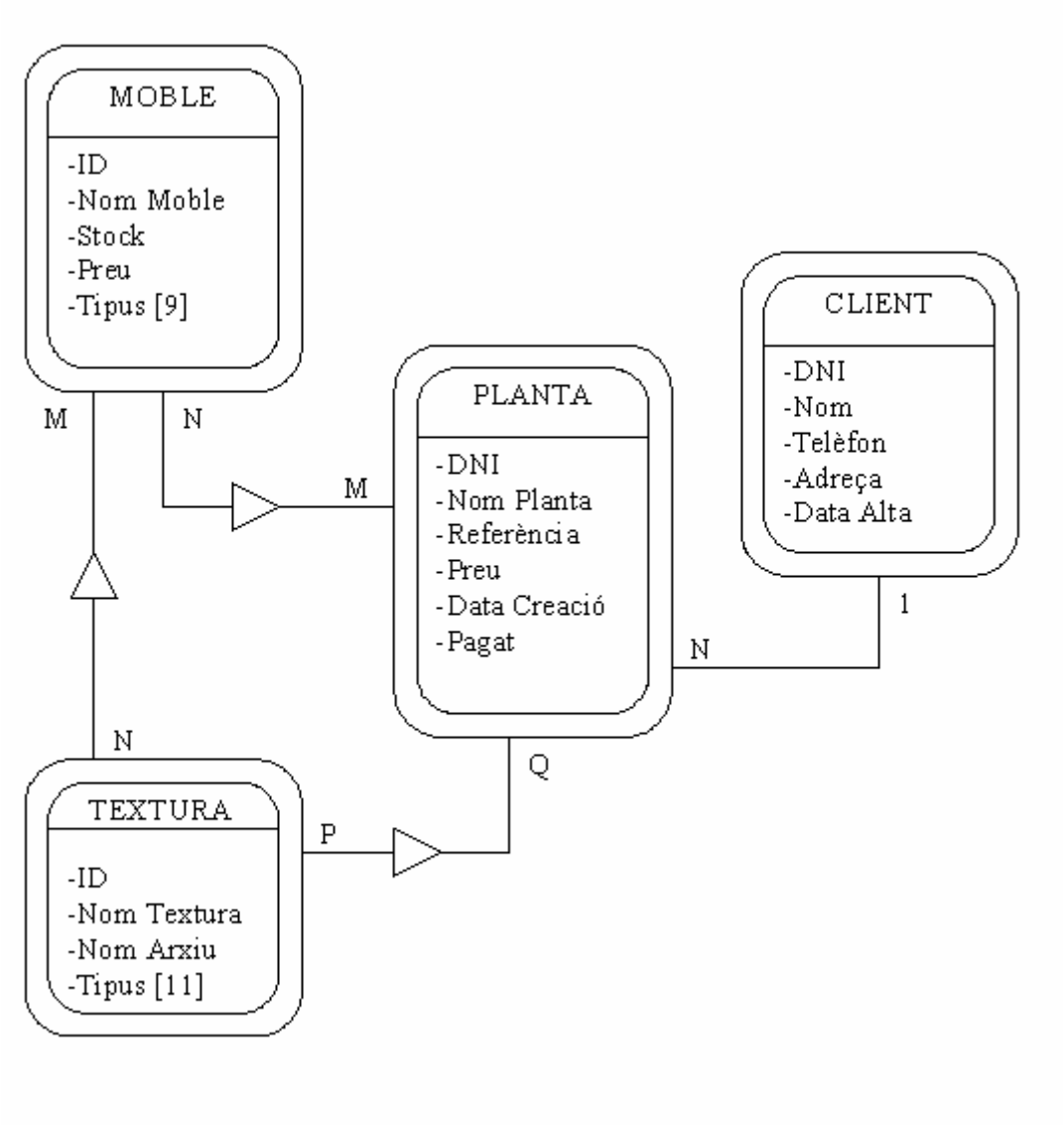
III.4.-Anàlisi

Ara utilitzarem Yourdon per tal de formalitzar l'especificació de requeriments.
Aquesta metodologia ens permet l'orientació a objectes, que ens facilitarà la comunicació amb l'usuari i evitar ambigüitats.
Hem d'explicar què fa el sistema: quina informació utilitza, quina d'aquesta informació ens interessa guardar i quines funcions ha de realitzar.
A continuació mostro el diagrama de classes, on hi veiem reflectits els objectes amb els seus atributs i la relació que hi ha entre ells.
Més endavant, ampliarem aquest diagrama per veure-hi totes les interfícies relacionades.

III.4.1.-Diagrama de classes:

El diagrama de classes el farem servir per mirar d'entendre com es mou la informació i quina és la que ens interessa guardar i començar a detallar els atributs que tindran els objectes descrits en el diagrama.

Cal recordar que el diagrama de classes ens donarà una visió una mica orientativa del que serà el sistema final.



III.4.2.-Descripció del diagrama de Classes

A continuació descric el diagrama de classes, mostrant quins atributs té cada objecte.

L'objecte **Moble** representa tots els mobles que tenim en stock a l'empresa.

Nom classe: MOBLES

Atributs:

- ID: int
- NOM MOBLE: *string*
- STOCK: int
- PREU: *int*
- CADIRA: string (bool)
- TAULA: string (bool)
- SOFÀ: string (bool)
- LLUM: string (bool)
- LLIT: string (bool)
- ARMARI: string (bool)
- CUINA: string (bool)
- LAVABO: string (bool)
- ALTRES: string (bool)

L'objecte **Textura** representa totes les textures que tenim al programa.

Nom classe: TEXTURA

Atributs:

- ID: int
- NOM TEXTURA: *string*
- NOM ARXIU:string
- TERRA: string (bool)
- PARET: string (bool)
- SOSTRE: string (bool)
- PORTA: string (bool)
- FINESTRA: string (bool)
- RESPALDO CADIRA: string (bool)
- POTES CADIRA: string (bool)
- BASE TAULA: string (bool)
- POTES TAULA: string (bool)
- ARMARI: string (bool)
- ASE: string (bool)

L'objecte **Planta** representa totes les plantes creades per l'empresa.

Nom classe: PLANTA
Atributs:
- DNI: *string*
- NOM PLANTA: *string*
- REFERENCIA: *string*
- PREU: string
- DATA CREACIO: COleDateTime
- PAGAT: string (bool)

L'objecte **Client** representa tots els clients que tenim a l'empresa.

Nom classe: CLIENT
Atributs:
- DNI: *string*
- NOM: *string*
- TELEFON: *string*
- ADREÇA: string
- DATA ALTA: COleDateTime

III.5.-Disseny

Una vegada acabada la fase d'anàlisi del sistema, passarem a la fase de disseny. Si l'objectiu de l'anàlisi era trobar el *que* ha de fer el futur sistema informàtic, en la fase de disseny l'objectiu és el *com* ho ha de fer. De manera que ara que ja sabem quina és la informació que hem de tractar, l'associarem als resultats obtinguts i a una tecnologia concreta.

El disseny constarà de la descripció dels tipus d'usuaris del sistema, els esdeveniments amb les seves respostes, el disseny d'interfícies de comunicació, l'ampliació del diagrama de classes amb les interfícies i els serveis de cada classe, el disseny de la base de dades, i per últim el disseny del programa.

III.5.1.-Tipus d'usuaris del sistema:

En aquesta part descriurem els usuaris que utilitzaran el sistema, tenint en compte les seves necessitats, i les seves restriccions.

El sistema només compta amb un tipus d'usuari, el dissenyador, tot i que cal tenir en compte que podria afegir-se'n un de manera molt senzilla, depenent de l'interès que tingués el propietari del sistema. És a dir, que el programa està pensat per a que el pugui fer servir qualsevol tipus d'usuari, però al tenir la base de dades oberta a tothom, descriurem només el dissenyador.

Dissenyador:

- **Tipus usuari:** Regular
- **Relació feina/sistema:** És, en principi, l'únic usuari del sistema, és l'encarregat tant d'organitzar les dades del client, com les de les habitacions, les textures, el mobles, les factures, llistats i el disseny d'habitacions.
- **Necessitats/Requeriments:** Cal que pugui accedir a tot arreu del programa, donar altes, baixes, modificacions i llistats de clients, altes, baixes, edicions i llistats d'habitacions, altes i consultes de textures, altes, edicions i consultes de mobles amb el seu stock, edicions i consultes de factures.
- **Forma de treballar:** El dissenyador és l'encarregat de suprvisar el sistema i portar-ho tot al dia, és el responsable de cobrar les factures pendents i de demanar més mobles en el cas de que el stock d'aquests sigui massa baix. I que sempre que vulgui pugui fer consultes sobre totes les dades, com veure el llistats de clients, habitacions, textures o mobles. No cal que hi consti ell al sistema.

III.5.2.-Especificació d'esdeveniments:

En aquesta secció, ens dedicarem a definir tots els esdeveniments que es poden dur a terme amb el projecte Base de Dades però també es esdeveniments que realitzen connexions amb l'ADO.

1. ***Nom esdeveniment:*** Es dóna d'alta un client.
   Tipus d'esdeveniment: Informació
   Resposta: Es registra un nou client
   Descripció: Es reben les dades del client i si no existeix, es registra.
   Serveis: client.buscar_client, client.alta_client.

2. ***Nom esdeveniment:*** Es dóna de baixa un client
   Tipus d'esdeveniment: Informació
   Resposta: Es destrueix un client
   Descripció: Es rep el dni del client i si existeix, s'esborren les dades, a més també s'esborren totes les habitacions assignades al client.
   Serveis: client.buscar_client, client.baixa_client, planta.baixa_totesplantes.

3. ***Nom esdeveniment:*** Es volen veure els clients
   Tipus d'esdeveniment: Informació
   Resposta: Es llisten tots els clients.
   Descripció: Es fa una consulta a tota la taula, mostrant els clients ordenades per ordre d'entrada.
   Serveis: Com que utilitzem un DataGridControl d'ActiveX Control (Microsoft) les llistes es fan directament des de l'arxiu .cpp, no des de l'objecte al que pertanyen.

4. ***Nom esdeveniment:*** Es vol modificar un client.
   Tipus d'esdeveniment: Informació
   Resposta: Es modificquen les dades del client.
   Descripció: Es rep el DNI del client que es vol modificar i si existeix es mostren les seves dades i el dissenyador les modifica, i es modifiquen a la Base de Dades.
   Serveis: client.buscar_client, client.mostrar_client, client.modificar_client.

5. ***Nom esdeveniment:*** Es dóna d'alta una habitació.
   Tipus d'esdeveniment: Informació
   Resposta: Es registra una nova habitació i es copia la seva referència al Planta_Actual.txt.
   Descripció: Es rep el dni del client, es registra i s'assigna al client en concret i es passa a dibuixar l'habitació.
   Serveis: client.buscar_client, planta.buscar_planta, planta.alta_planta.

6. ***Nom esdeveniment:*** Es dóna de baixa una habitació.
   Tipus d'esdeveniment: Informació
   Resposta: Es destrueix una habitació.
   Descripció: Es rep el dni o el nom del client i si existeix, es mostren les habitacions que té assignades el client, es rep el nom de l'habitació sel·leccionada i es destrueix.
   Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, planta.baixa_planta.

7. ***Nom esdeveniment:*** Es dóna de totes les habitacions d'un client.
   Tipus d'esdeveniment: Informació
   Resposta: Es destrueix una habitació.
   Descripció: Es rep el dni o el nom del client i si existeix, es mostren les habitacions que té assignades el client i es destrueixen totes aquestes habitacions.
   Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, planta.baixa_totesplantes.

8. ***Nom esdeveniment:*** Es volen veure les habitacions
   Tipus d'esdeveniment: Informació
   Resposta: Es llisten totes les habitacions.
   Descripció: Es fa una consulta a tota la taula, mostrant les habitacions ordenades per client, que està ordenat per ordre d'entrada.
   Serveis: Com que utilitzem un DataGridControl d'ActiveX Control (Microsoft) les llistes es fan directament des de l'arxiu .cpp, no des de l'objecte al que pertanyen.

9. ***Nom esdeveniment:*** Es vol obrir una habitació per editar-la.
   Tipus d'esdeveniment: Informació
   Resposta: Es reedita una habitació i es copia la seva referència al Planta_Actual.txt.
   Descripció: Es rep el dni o el nom del client i si existeix, es mostren les habitacions que té assignades el client, es rep el nom de l'habitació en concret que es vol obrir i s'obre, això et porta a l'Editor d'Habitacions.
   Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista.

10. ***Nom esdeveniment:*** Es dóna d'alta una textura.
    Tipus d'esdeveniment: Informació
    Resposta: Es registra una nova textura i es copia la seva imatge a la carpeta textures.
    Descripció: Es rep el nom de la textura amb els tipus de superfícies als que pertany, i si no existeix, s'obre el carregador es rep el path de la textura que es vol donar d'alta i es fa la còpia a la carpeta Docs\Textures\nom_textura.bmp.
    Serveis: textura.buscar_textura, textura.alta_textura.

11. ***Nom esdeveniment:*** Es volen veure les textures.
    Tipus d'esdeveniment: Informació
    Resposta: Es llisten totes les textures.
    Descripció: Es fa una consulta a tota la taula, mostrant les textures ordenades per ordre d'entrada.
    Serveis: Com que utilitzem un DataGridControl d'ActiveX Control (Microsoft) les llistes es fan directament des de l'arxiu .cpp, no des de l'objecte al que pertanyen.

12. ***Nom esdeveniment:*** Es dóna d'alta un moble.
    Tipus d'esdeveniment: Informació
    Resposta: Es registra un nou moble i es copia el .ASE i el .txt a la carpeta ASE.
    Descripció: Es rep el nom del moble amb el preu i el stock amb els tipus de mobles als que pertany, i si no existeix, s'obre el carregador es rep el path del moble (.ASE) que es vol donar d'alta, el programa interpreta l'arxiu .ASE i el transforma a .txt, es fa la còpia a la carpeta Docs\ASE\nom_moble.ASE i es crea el nou arxiu .txt a la carpeta Docs\ASE\nom_moble.txt.
    .
    Serveis: moble.buscar_Idmoble, moble.buscar_moble, moble.alta_moble.

13. ***Nom esdeveniment:*** Es vol modificar el stock d'un moble.
Tipus d'esdeveniment: Informació
Resposta: Es modifica el stock d'un moble.
Descripció: Es rep el nom del moble que es vol modificar el stock, si existeix, mostra el stock del moble en concret, es rep la nova xifra i és modifica al base de dades.
Serveis: moble.buscar_moble, moble.mostrar_infoMoble, moble.modificar_moble.

14. ***Nom esdeveniment:*** Es vol modificar el preu d'un moble.
Tipus d'esdeveniment: Informació
Resposta: Es modifica el preu d'un moble.
Descripció: Es rep el nom del moble que es vol modificar el preu, si existeix, mostra el preu del moble en concret, es rep la nova xifra i és modifica al base de dades.
Serveis: moble.buscar_moble, moble.mostrar_infoMoble, moble.modificar_moble.

15. ***Nom esdeveniment:*** Es volen veure les factures pendents.
Tipus d'esdeveniment: Informació
Resposta: Es llisten les factures que falten per pagar.
Descripció: Es fa una consulta a la base de dades i es mostren les factures de totes les plantes que queden pendents de pagar.
Serveis: planta.mostrar_nopagat.

16. ***Nom esdeveniment:*** Es volen veure les factures pendents d'un client.
Tipus d'esdeveniment: Informació
Resposta: Es mostren les factures que falten per pagar d'un client.
Descripció: Es fa una consulta a la base de dades i es mostren les factures de totes les plantes que queden pendents de pagar, se sel·lecciona la factura en concret i et retorna les dades del client que falta per pagar.
Serveis: planta.mostrar_nopagat, planta.retorna_dni_factura, client.mostrar_client.

17. ***Nom esdeveniment:*** Es volen veure la factura detallada d'una habitació.
Tipus d'esdeveniment: Informació
Resposta: Es mostren les dades de tots els mobles que conté l'habitació.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra el llistat de mobles de l'habitació, i si se sel·lecciona qualsevol moble de la llista es mostren les dades de preu i stock, també es mostra el preu total, l'IVA total i el total+IVA.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, planta.buscar_pagat. moble.mostrar_infoMoble, ///moble.Generar_Factura.//// moble.restar_stock //// planta.marcar_pagat

18. ***Nom esdeveniment:*** Es vol generar una factura.
Tipus d'esdeveniment: Informació
Resposta: Es mostren les dades de tots els mobles que conté l'habitació, es genera factura i es resta stock.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra el llistat de mobles de l'habitació, i si se sel·lecciona qualsevol moble de la llista es mostren les dades de preu i stock, també es mostra el preu total, l'IVA total i el total+IVA, es pregunta si es vol generar la factura, si és que si es resten els mobles que conté l'habitació del stock i es genera la factura.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, planta.buscar_pagat, moble.Generar_Factura.

19. ***Nom esdeveniment:*** Es vol pagar una factura detallada.
Tipus d'esdeveniment: Informació
Resposta: Es paga factura.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra el llistat de mobles de l'habitació, i si se sel·lecciona qualsevol moble de la llista es mostren les dades de preu i stock, també es mostra el preu total, l'IVA total i el total+IVA, si la factura no està generada es pregunta si es vol generar la factura, si és que si es resten els mobles que conté l'habitació del stock i es genera la factura i es pregunta si es vol pagar, si és que sí la factura constarà com a pagada.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, planta.buscar_pagat, moble.Generar_Factura, moble.restar_stock, planta.marcar_pagat.

20. ***Nom esdeveniment:*** Es vol consultar l'stock.
Tipus d'esdeveniment: Informació
Resposta: Es mostren els mobles amb poc stock.
Descripció: Es fa una consulta a la base de dades dels mobles que tenen stock inferior a 5, a 10, a 20 o a 30 i es mostren els mobles da cada grup d'aquests.
Serveis: moble.mostrar_consulta_stock.

21. ***Nom esdeveniment:*** Es vol editar una habitació.
Tipus d'esdeveniment: Informació
Resposta: Es mostra l'Editor d'Habitacions.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra l'Editor d'Habitacions.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, Buscar_numIDs, Buscar_IDTextura, Buscar_NomASE.

22. ***Nom esdeveniment:*** Es vol fer una visita interactiva d'una habitació.
Tipus d'esdeveniment: Informació
Resposta: Es mostra el Visita Interactiva.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra el Visita Interactiva.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, Buscar_numIDs, Buscar_IDTextura.

23. ***Nom esdeveniment:*** Es vol veure una habitació des de diferents punts de vista.
Tipus d'esdeveniment: Informació
Resposta: Es mostra el Vistes.
Descripció: Es rep el dni o el nom del client, es mostren les habitacions que té, es rep el nom de l'habitació sel·leccionada, s'obre, es mostra el Vistes.
Serveis: client.buscar_client, client.buscar_client_nom, planta.mostrar_llista, Buscar_numIDs, Buscar_IDTextura, Buscar_NomASE.

24. ***Nom esdeveniment:*** Es vol veure l'ajuda.
Tipus d'esdeveniment: Informació
Resposta: Es mostra l'ajuda.
Descripció: Es rep la petició d'ajuda i es mostra l'ajuda en html.
Serveis:

25. ***Nom esdeveniment:*** Es vol sortir de l'aplicació.
Tipus d'esdeveniment: Informació
Resposta: Surt de l'aplicació.
Descripció: Es rep la petició de sortir de l'aplicació i surt.
Serveis:

26. ***Nom esdeveniment:*** Es llistar un tipus de moble stàndar.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat del tipus triat de moble stàndar.
Descripció: Dintre de l'Editor d'Habitacions, al afegir un moble stàndar, mostra una llista amb tots els tipus de mobles stàndar, se sel·lecciona un i es mostra llistat dels mobles stàndar que pertanyen al moble sel·leccionat.
Serveis: mostrar_mobles.

27. ***Nom esdeveniment:*** Es vol dibuixar una paret.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat de textures del tipus paret.
Descripció: Dintre del Dibuixar Habitacions, al sel·leccionar una paret mostra un llistat de totes les textures de tipus paret.
Serveis: mostrar_text, Buscar_IDTextura.

28. ***Nom esdeveniment:*** Es vol dibuixar una porta.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat de textures del tipus porta.
Descripció: Dintre del Dibuixar Habitacions, al sel·leccionar una porta mostra un llistat de totes les textures de tipus porta.
Serveis: mostrar_text, Buscar_IDTextura.

29. ***Nom esdeveniment:*** Es vol dibuixar una finestra.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat de textures del tipus finestra.
Descripció: Dintre del Dibuixar Habitacions, al sel·leccionar una finestra mostra un llistat de totes les textures de tipus finestra.
Serveis: mostrar_text, Buscar_IDTextura.

30. ***Nom esdeveniment:*** Es vol dibuixar el terra de l'habitació.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat de textures del tipus terra.
Descripció: Dintre del Dibuixar Habitacions, al sel·leccionar una terra mostra un llistat de totes les textures de tipus terra.
Serveis: mostrar_text, Buscar_IDTextura.

31. ***Nom esdeveniment:*** Es vol dibuixar el sostre de l'habitació.
Tipus d'esdeveniment: Informació
Resposta: Es mostra llistat de textures del tipus sostre.
Descripció: Dintre del Dibuixar Habitacions, al sel·leccionar una sostre mostra un llistat de totes les textures de tipus sostre.
Serveis: mostrar_text, Buscar_IDTextura.

32. ***Nom esdeveniment:*** Es vol crear una cadira a mida.
Tipus d'esdeveniment: Informació
Resposta: Es mostren 2 llistats de textures, el de tipus cadira respaldo i el del tipus cadira estructura.
Descripció: Dintre de l'Editor d'Habitacions, a l'afegir un moble a mida, sel·leccionem cadira i es mostren 2 llistats de textures, el del respaldo i el de l'estructura.
Serveis: mostrar_text, Buscar_IDTextura.

33. ***Nom esdeveniment:*** Es vol crear una taula a mida.

Tipus d'esdeveniment: Informació

Resposta: Es mostren 2 llistats de textures, el de tipus taula potes i el del tipus taula taula.

Descripció: Dintre de l'Editor d'Habitacions, a l'afegir un moble a mida, sel·leccionem taula i es mostren 2 llistats de textures, el de les potes i el de la taula.

Serveis: mostrar_text, Buscar_IDTextura.

34. ***Nom esdeveniment:*** Es vol crear un armari a mida.

Tipus d'esdeveniment: Informació

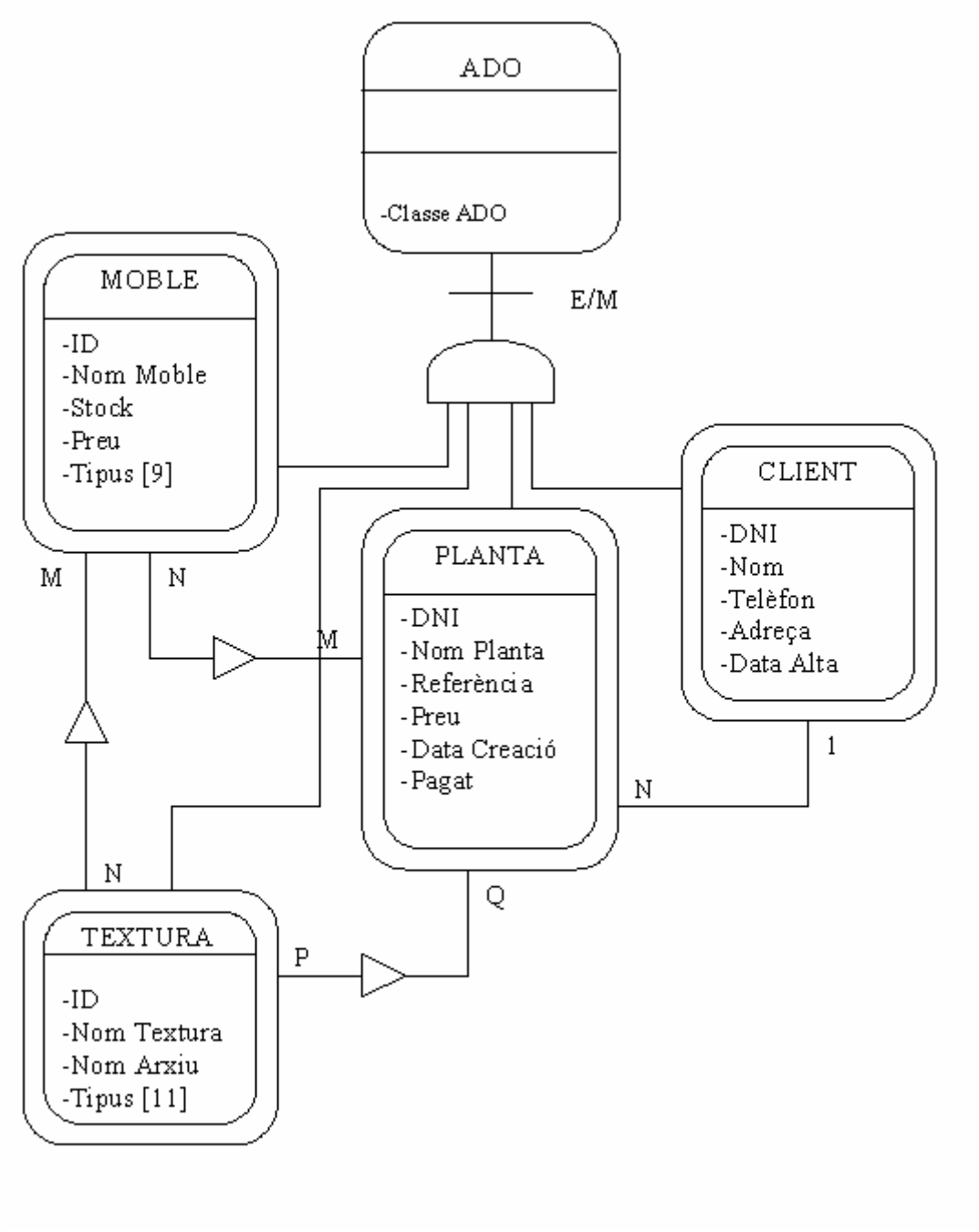
Resposta: Es mostra llistat de textures de tipus armari.

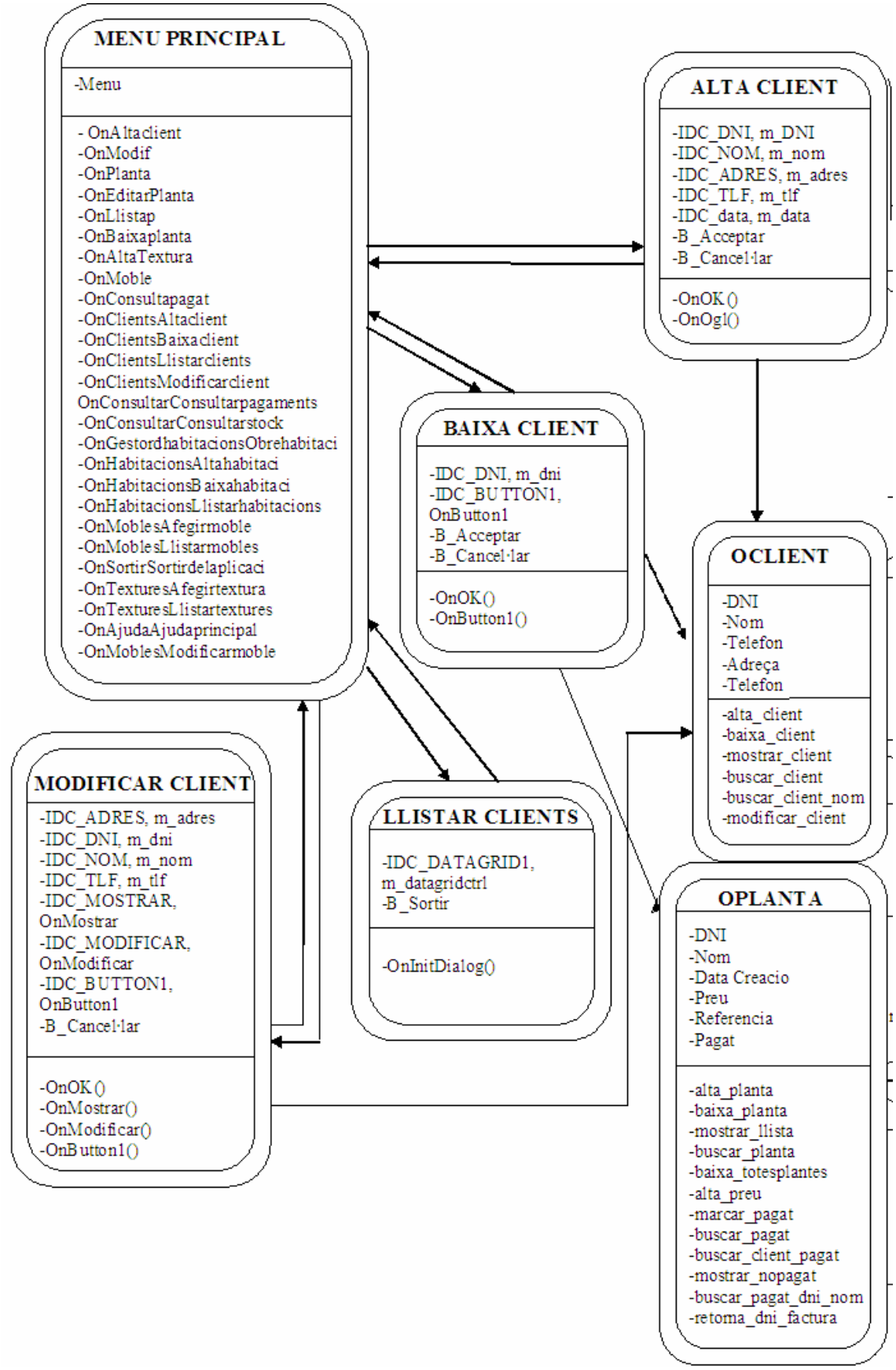
Descripció: Dintre de l'Editor d'Habitacions, a l'afegir un moble a mida, sel·leccionem armari i es mostra el llistat de textures de tipus armari.

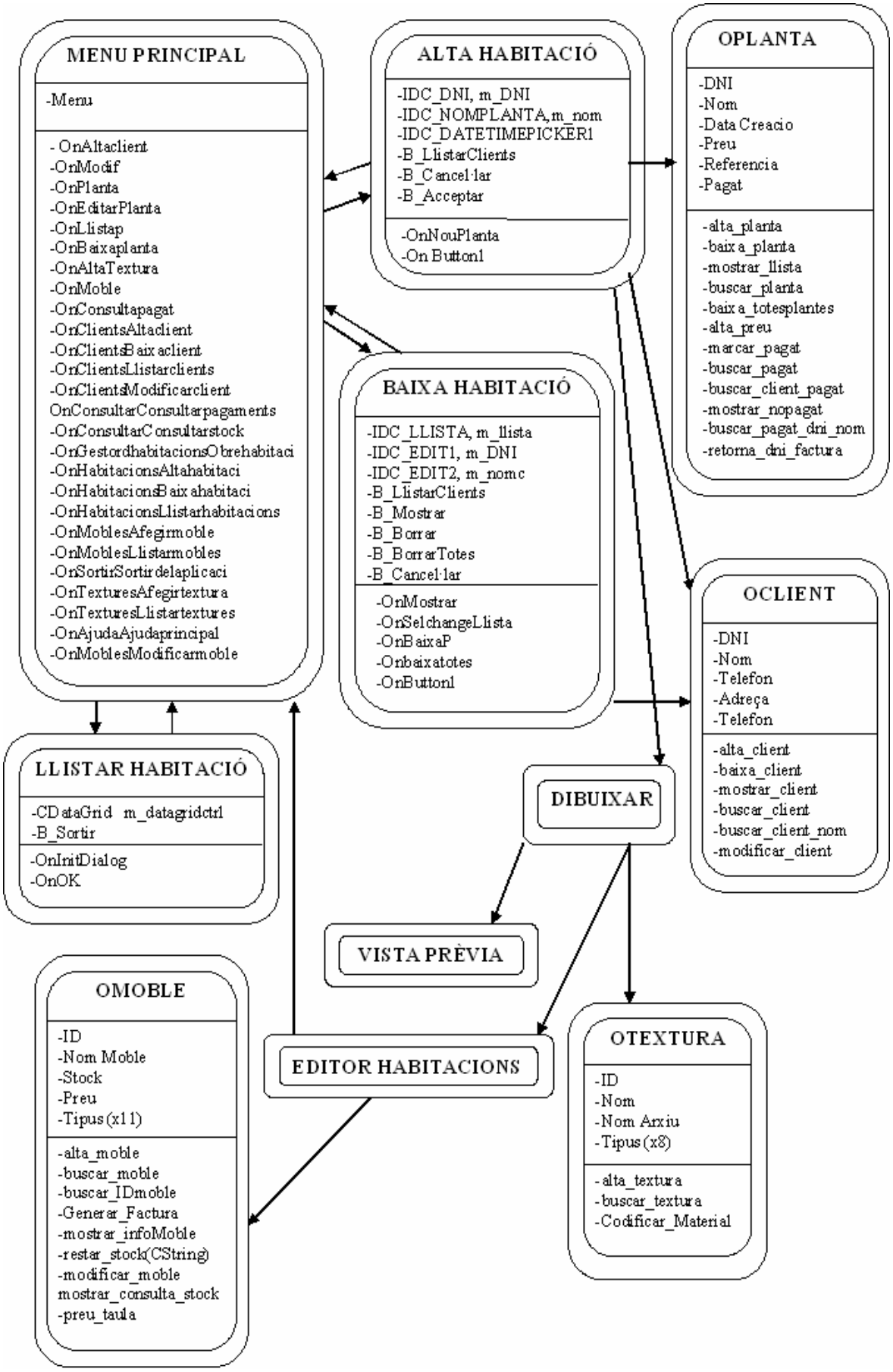
Serveis: mostrar_text, Buscar_IDTextura.

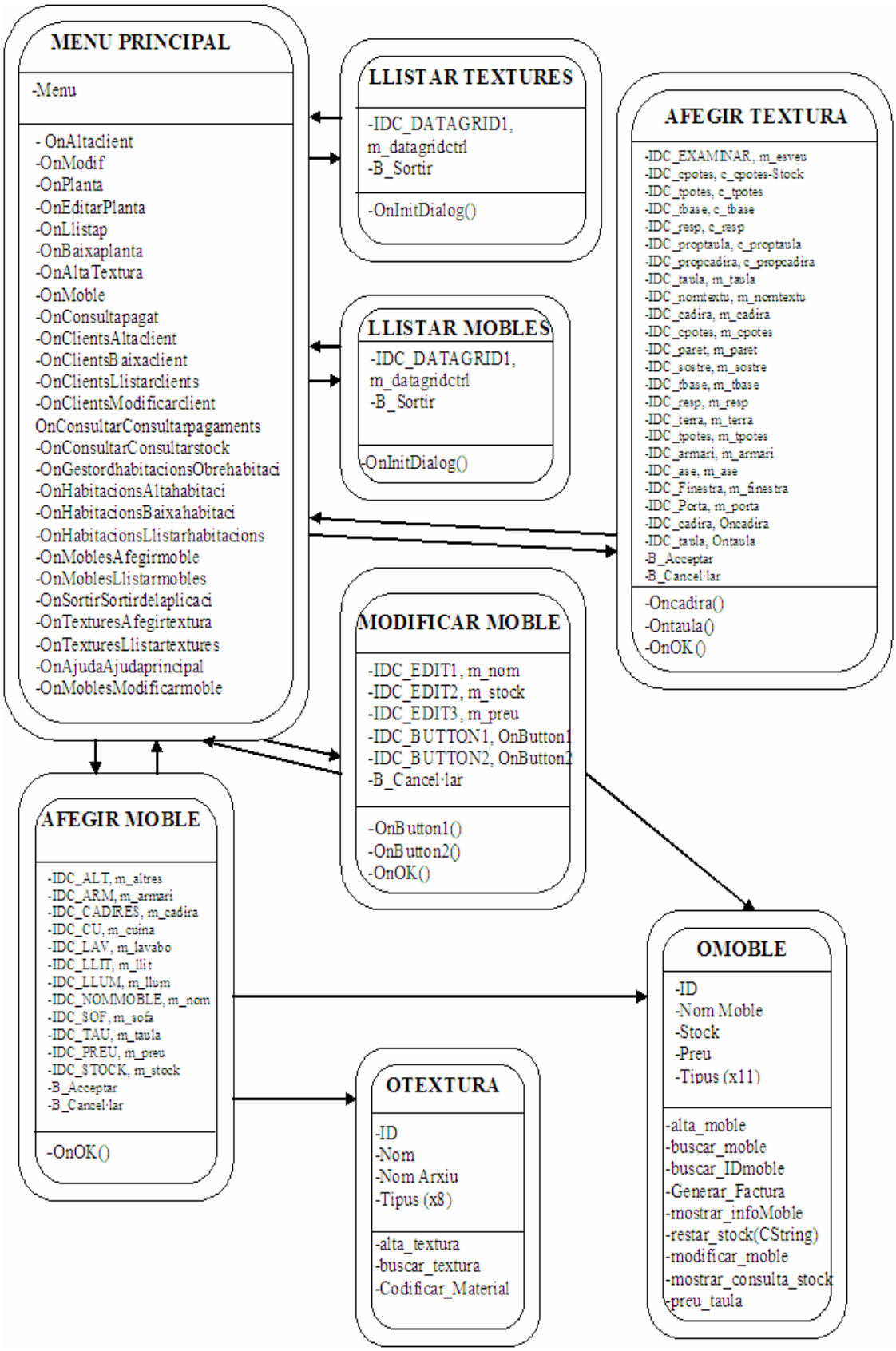
III.5.3.-Ampliació del diagrama de classes:

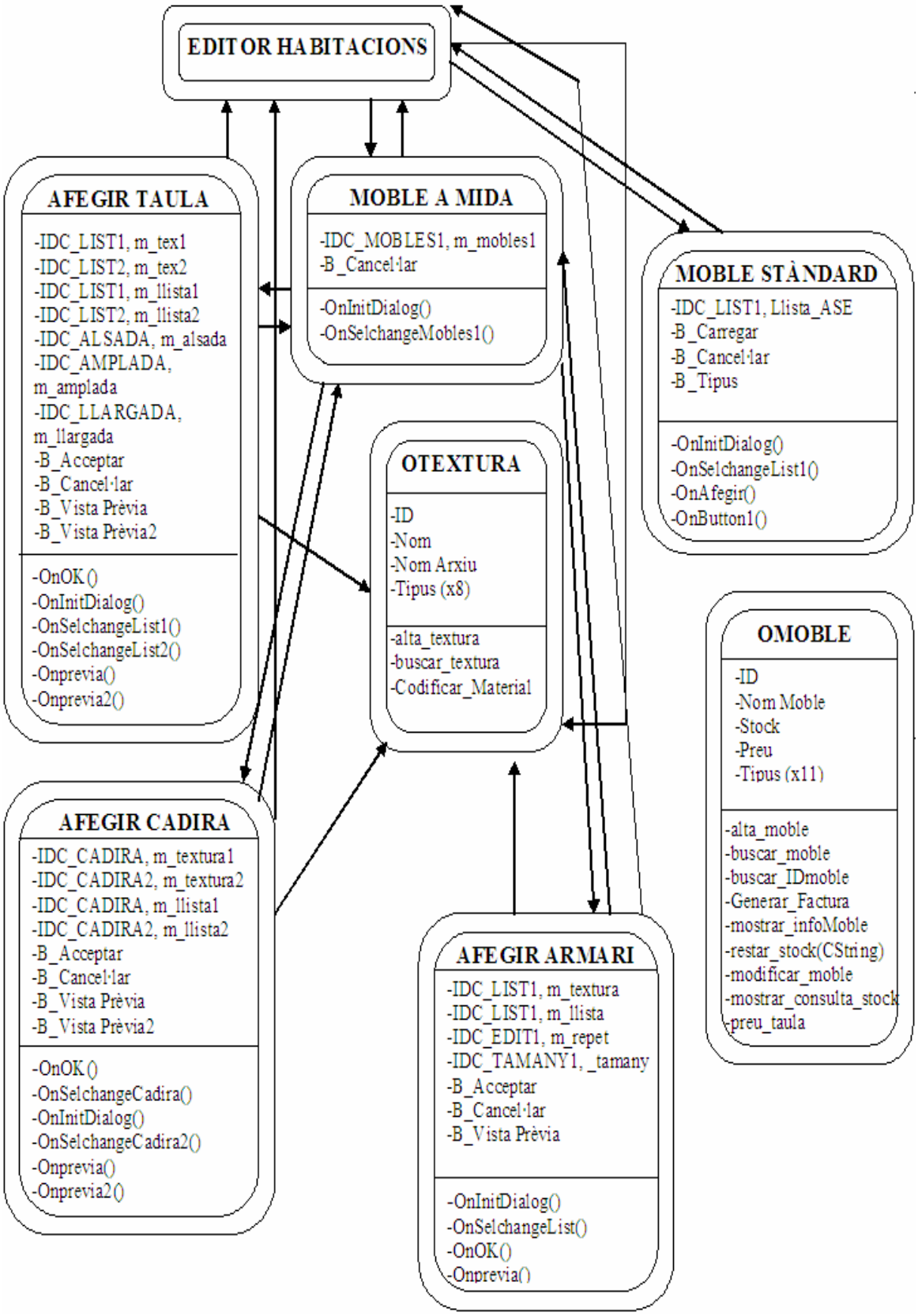
El diagrama de classes ampliat està format pel diagrama de classes que ja tenim més les interfícies relacionades. A més també hi afegirem una classe nova,l'ADOl. Que és una classe abstacta perquè només conté aplicacions bàsiques utilitzades per tots els altres objectes.

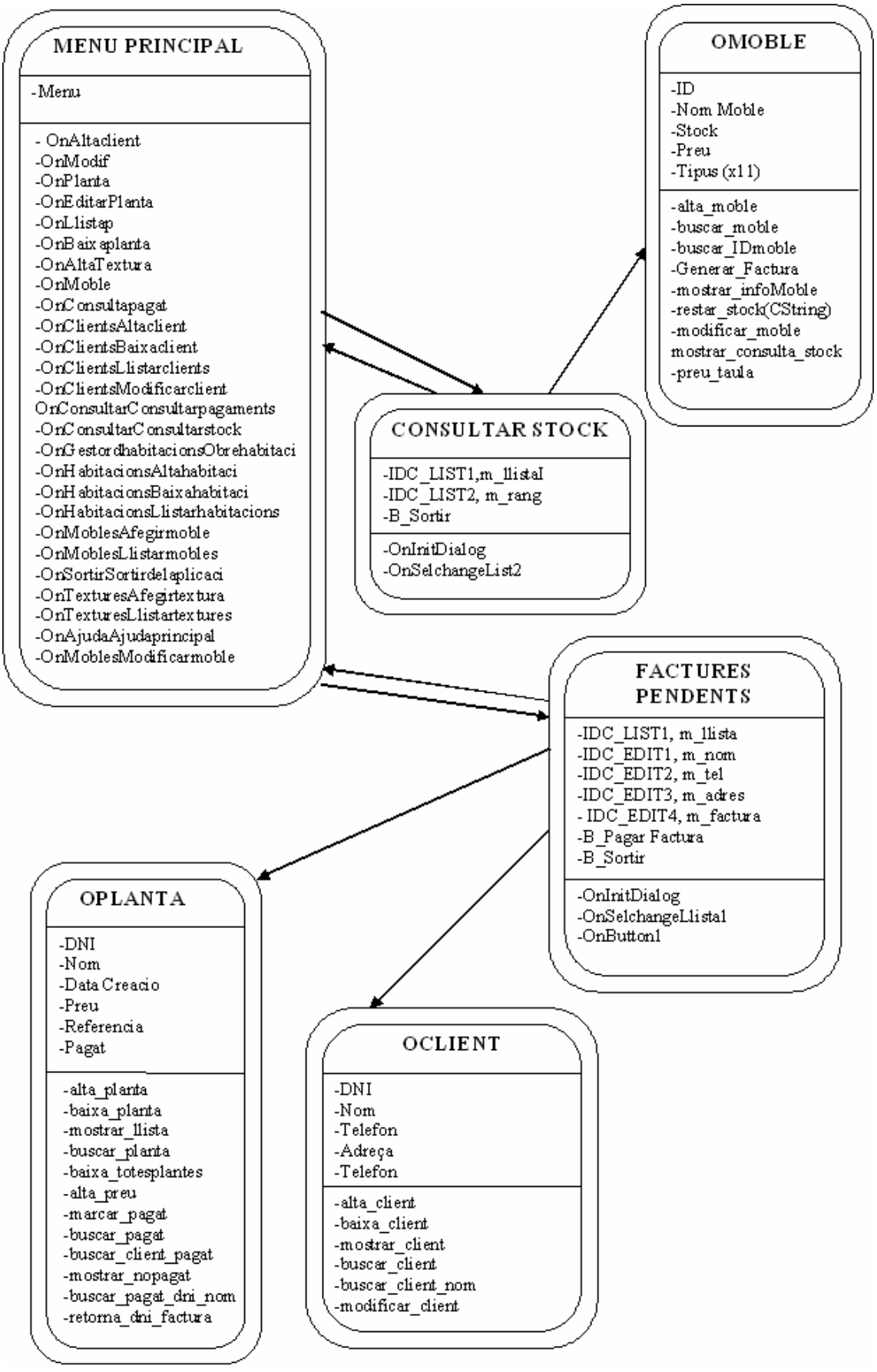


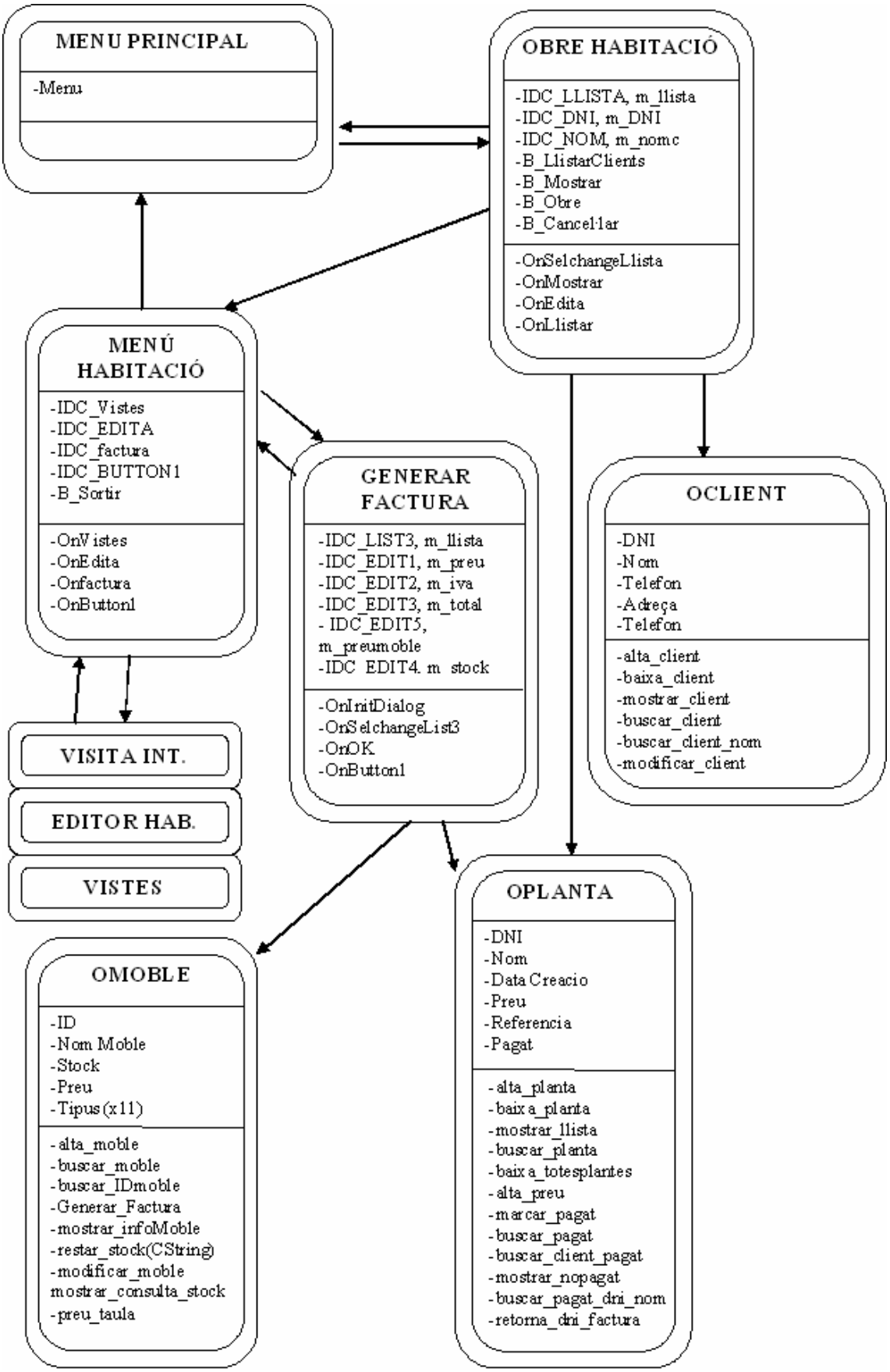

MENU PRINCIPAL
-Menu
- OnAltaclient
-OnModif
-OnPlanta
-OnEditarPlanta
-OnLlistap
-OnBaixaplanta
-OnAltaTextura
-OnMoble
-OnConsultapagat
-OnClientsAltaclient
-OnClientsBaixaclient
-OnClientsLlistarclients
-OnClientsModificarclient
OnConsultarConsultarpagaments
-OnConsultarConsultarstock
-OnGestordhabitacionsObrehabitaci
-OnHabitacionsAltahabitaci
-OnHabitacionsBaixahabitaci
-OnHabitacionsLlistarhabitacions
-OnMoblesAfegirmoble
-OnMoblesLlistarmobles
-OnSortirSortirdelaplicaci
-OnTexturesAfegirtextura
-OnTexturesLlistartextures
-OnAjudaAjudaprincipal
-OnMoblesModificarmoble
ALTA CLIENT
-IDC_DNI, m_DNI
-IDC_NOM, m_nom
-IDC_ADRES, m_adres
-IDC_TLF, m_tlf
-IDC_data, m_data
-B_Acceptar
-B_Cancel·lar
-OnOK()
-OnOgl()
BAIXA CLIENT
-IDC_DNI, m_dni
-IDC_BUTTON1,
OnButton1
-B_Acceptar
-B_Cancel·lar
-OnOK()
-OnButton1()
OCLIENT
-DNI
-Nom
-Telefon
-Adreça
-Telefon
-alta_client
-baixa_client
-mostrar_client
-buscar_client
-buscar_client_nom
-modificar_client
MODIFICAR CLIENT
-IDC_ADRES, m_adres
-IDC_DNI, m_dni
-IDC_NOM, m_nom
-IDC_TLF, m_tlf
-IDC_MOSTRAR,
OnMostrar
-IDC_MODIFICAR,
OnModificar
-IDC_BUTTON1,
OnButton1
-B_Cancel·lar
-OnOK()
-OnMostrar()
-OnModificar()
-OnButton1()
LLISTAR CLIENTS
-IDC_DATAGRID1,
m_datagridctrl
-B_Sortir
-OnInitDialog()
OPLANTA
-DNI
-Nom
-Data Creacio
-Preu
-Referencia
-Pagat
-alta_planta
-baixa_planta
-mostrar_llista
-buscar_planta
-baixa_totesplantes
-alta_preu
-marcar_pagat
-buscar_pagat
-buscar_client_pagat
-mostrar_nopagat
-buscar_pagat_dni_nom
-retorna_dni_factura


MENU PRINCIPAL
-Menu
- OnAltaclient
-OnModif
-OnPlanta
-OnEditarPlanta
-OnLlistap
-OnBaixaplanta
-OnAltaTextura
-OnMoble
-OnConsultapagat
-OnClientsAltaclient
-OnClientsBaixaclient
-OnClientsLlistarclients
-OnClientsModificarclient
OnConsultarConsultarpagaments
-OnConsultarConsultarstock
-OnGestordhabitacionsObrehabitaci
-OnHabitacionsAltahabitaci
-OnHabitacionsBaixahabitaci
-OnHabitacionsLlistarhabitacions
-OnMoblesAfegirmoble
-OnMoblesLlistarmobles
-OnSortirSortirdelaplicaci
-OnTexturesAfegirtextura
-OnTexturesLlistartextures
-OnAjudaAjudaprincipal
-OnMoblesModificarmoble
ALTA HABITACIÓ
-IDC_DNI, m_DNI
-IDC_NOMPLANTA,m_nom
-IDC_DATETIMEPICKER1
-B_LlistarClients
-B_Cancel·lar
-B_Acceptar
-OnNouPlanta
-On Button1
OPLANTA
-DNI
-Nom
-Data Creacio
-Preu
-Referencia
-Pagat
-alta_planta
-baixa_planta
-mostrar_llista
-buscar_planta
-baixa_totesplantes
-alta_preu
-marcar_pagat
-buscar_pagat
-buscar_client_pagat
-mostrar_nopagat
-buscar_pagat_dni_nom
-retorna_dni_factura
BAIXA HABITACIÓ
-IDC_LLISTA, m_llista
-IDC_EDIT1, m_DNI
-IDC_EDIT2, m_nomc
-B_LlistarClients
-B_Mostrar
-B_Borrar
-B_BorrarTotes
-B_Cancel·lar
-OnMostrar
-OnSelchangeLlista
-OnBaixaP
-Onbaixatotes
-OnButton1
OCLIENT
-DNI
-Nom
-Telefon
-Adreça
-Telefon
-alta_client
-baixa_client
-mostrar_client
-buscar_client
-buscar_client_nom
-modificar_client
LLISTAR HABITACIÓ
-CDataGrid m_datagridctrl
-B_Sortir
-OnInitDialog
-OnOK
DIBUIXAR
VISTA PRÈVIA
OMOBLE
-ID
-Nom Moble
-Stock
-Preu
-Tipus(x11)
-alta_moble
-buscar_moble
-buscar_IDmoble
-Generar_Factura
-mostrar_infoMoble
-restar_stock(CString)
-modificar_moble
mostrar_consulta_stock
-preu_taula
EDITOR HABITACIONS
OTEXTURA
-ID
-Nom
-Nom Arxiu
-Tipus(x8)
-alta_textura
-buscar_textura
-Codificar_Material


MENU PRINCIPAL
-Menu
- OnAltaclient
-OnModif
-OnPlanta
-OnEditarPlanta
-OnLlistap
-OnBaixaplanta
-OnAltaTextura
-OnMoble
-OnConsultapagat
-OnClientsAltaclient
-OnClientsBaixaclient
-OnClientsLlistarclients
-OnClientsModificarclient
OnConsultarConsultarpagaments
-OnConsultarConsultarstock
-OnGestordhabitacionsObrehabitaci
-OnHabitacionsAltahabitaci
-OnHabitacionsBaixahabitaci
-OnHabitacionsLlistarhabitacions
-OnMoblesAfegirmoble
-OnMoblesLlistarmobles
-OnSortirSortirdelaplicaci
-OnTexturesAfegirtextura
-OnTexturesLlistartextures
-OnAjudaAjudaprincipal
-OnMoblesModificarmoble
LLISTAR TEXTURES
-IDC_DATAGRID1, m_datagridctrl
-B_Sortir
-OnInitDialog()
LLISTAR MOBLES
-IDC_DATAGRID1, m_datagridctrl
-B_Sortir
-OnInitDialog()
AFEGIR TEXTURA
-IDC_EXAMINAR, m_esveu
-IDC_cpotes, c_cpotes-Stock
-IDC_tpotes, c_tpotes
-IDC_tbase, c_tbase
-IDC_resp, c_resp
-IDC_proptaula, c_proptaula
-IDC_propcadira, c_propcadira
-IDC_taula, m_taula
-IDC_nomtextu, m_nomtextu
-IDC_cadira, m_cadira
-IDC_cpotes, m_cpotes
-IDC_paret, m_paret
-IDC_sostre, m_sostre
-IDC_tbase, m_tbase
-IDC_resp, m_resp
-IDC_terra, m_terra
-IDC_tpotes, m_tpotes
-IDC_armari, m_armari
-IDC_ase, m_ase
-IDC_Finestra, m_finestra
-IDC_Porta, m_porta
-IDC_cadira, Oncadira
-IDC_taula, Ontaula
-B_Acceptar
-B_Cancel·lar
-Oncadira()
-Ontaula()
-OnOK()
MODIFICAR MOBLE
-IDC_EDIT1, m_nom
-IDC_EDIT2, m_stock
-IDC_EDIT3, m_preu
-IDC_BUTTON1, OnButton1
-IDC_BUTTON2, OnButton2
-B_Cancel·lar
-OnButton1()
-OnButton2()
-OnOK()
AFEGIR MOBLE
-IDC_ALT, m_altres
-IDC_ARM, m_armari
-IDC_CADIRES, m_cadira
-IDC_CU, m_cuina
-IDC_LAV, m_lavabo
-IDC_LLIT, m_llit
-IDC_LLUM, m_llum
-IDC_NOMMOBLE, m_nom
-IDC_SOF, m_sofa
-IDC_TAU, m_taula
-IDC_PREU, m_preu
-IDC_STOCK, m_stock
-B_Acceptar
-B_Cancel·lar
-OnOK()
OMOBLE
-ID
-Nom Moble
-Stock
-Preu
-Tipus (x11)
-alta_moble
-buscar_moble
-buscar_IDmoble
-Generar_Factura
-mostrar_infoMoble
-restar_stock(CString)
-modificar_moble
-mostrar_consulta_stock
-preu_taula
OTEXTURA
-ID
-Nom
-Nom Arxiu
-Tipus (x8)
-alta_textura
-buscar_textura
-Codificar_Material


EDITOR HABITACIONS
AFEGIR TAULA
-IDC_LIST1, m_tex1
-IDC_LIST2, m_tex2
-IDC_LIST1, m_llista1
-IDC_LIST2, m_llista2
-IDC_ALSADA, m_alsada
-IDC_AMPLADA, m_amplada
-IDC_LLARGADA, m_llargada
-B_Acceptar
-B_Cancel·lar
-B_Vista Prèvia
-B_Vista Prèvia2
-OnOK()
-OnInitDialog()
-OnSelchangeList1()
-OnSelchangeList2()
-Onprevia()
-Onprevia2()
MOBLE A MIDA
-IDC_MOBLES1, m_mobles1
-B_Cancel·lar
-OnInitDialog()
-OnSelchangeMobles1()
MOBLE STÀNDARD
-IDC_LIST1, Llista_ASE
-B_Carregar
-B_Cancel·lar
-B_Tipus
-OnInitDialog()
-OnSelchangeList1()
-OnAfegir()
-OnButton1()
OTEXTURA
-ID
-Nom
-Nom Arxiu
-Tipus (x8)
-alta_textura
-buscar_textura
-Codificar_Material
OMOBLE
-ID
-Nom Moble
-Stock
-Preu
-Tipus (x11)
-alta_moble
-buscar_moble
-buscar_IDmoble
-Generar_Factura
-mostrar_infoMoble
-restar_stock(CString)
-modificar_moble
-mostrar_consulta_stock
-preu_taula
AFEGIR CADIRA
-IDC_CADIRA, m_textura1
-IDC_CADIRA2, m_textura2
-IDC_CADIRA, m_llista1
-IDC_CADIRA2, m_llista2
-B_Acceptar
-B_Cancel·lar
-B_Vista Prèvia
-B_Vista Prèvia2
-OnOK()
-OnSelchangeCadira()
-OnInitDialog()
-OnSelchangeCadira2()
-Onprevia()
-Onprevia2()
AFEGIR ARMARI
-IDC_LIST1, m_textura
-IDC_LIST1, m_llista
-IDC_EDIT1, m_repet
-IDC_TAMANY1, _tamany
-B_Acceptar
-B_Cancel·lar
-B_Vista Prèvia
-OnInitDialog()
-OnSelchangeList()
-OnOK()
-Onprevia()


MENU PRINCIPAL
-Menu
- OnAltaclient
-OnModif
-OnPlanta
-OnEditarPlanta
-OnLlistap
-OnBaixaplanta
-OnAltaTextura
-OnMoble
-OnConsultapagat
-OnClientsAltaclient
-OnClientsBaixaclient
-OnClientsLlistarclients
-OnClientsModificarclient
OnConsultarConsultarpagaments
-OnConsultarConsultarstock
-OnGestordhabitacionsObrehabitaci
-OnHabitacionsAltahabitaci
-OnHabitacionsBaixahabitaci
-OnHabitacionsLlistarhabitacions
-OnMoblesAfegirmoble
-OnMoblesLlistarmobles
-OnSortirSortirdelaplicaci
-OnTexturesAfegirtextura
-OnTexturesLlistartextures
-OnAjudaAjudaprincipal
-OnMoblesModificarmoble
OMOBLE
-ID
-Nom Moble
-Stock
-Preu
-Tipus (x11)
-alta_moble
-buscar_moble
-buscar_IDmoble
-Generar_Factura
-mostrar_infoMoble
-restar_stock(CString)
-modificar_moble
mostrar_consulta_stock
-preu_taula
CONSULTAR STOCK
-IDC_LIST1,m_llistal
-IDC_LIST2, m_rang
-B_Sortir
-OnInitDialog
-OnSelchangeList2
FACTURES PENDENTS
-IDC_LIST1, m_llista
-IDC_EDIT1, m_nom
-IDC_EDIT2, m_tel
-IDC_EDIT3, m_adres
- IDC_EDIT4, m_factura
-B_Pagar Factura
-B_Sortir
-OnInitDialog
-OnSelchangeLlistal
-OnButton1
OPLANTA
-DNI
-Nom
-Data Creacio
-Preu
-Referencia
-Pagat
-alta_planta
-baixa_planta
-mostrar_llista
-buscar_planta
-baixa_totesplantes
-alta_preu
-marcar_pagat
-buscar_pagat
-buscar_client_pagat
-mostrar_nopagat
-buscar_pagat_dni_nom
-retorna_dni_factura
OCLIENT
-DNI
-Nom
-Telefon
-Adreça
-Telefon
-alta_client
-baixa_client
-mostrar_client
-buscar_client
-buscar_client_nom
-modificar_client


MENU PRINCIPAL
-Menu
OBRE HABITACIÓ
-IDC_LLISTA, m_llista
-IDC_DNI, m_DNI
-IDC_NOM, m_nomc
-B_LlistarClients
-B_Mostrar
-B_Obre
-B_Cancel·lar
-OnSelchangeLlista
-OnMostrar
-OnEdita
-OnLlistar
MENÚ HABITACIÓ
-IDC_Vistes
-IDC_EDITA
-IDC_factura
-IDC_BUTTON1
-B_Sortir
-OnVistes
-OnEdita
-Onfactura
-OnButton1
GENERAR FACTURA
-IDC_LIST3, m_llista
-IDC_EDIT1, m_preu
-IDC_EDIT2, m_iva
-IDC_EDIT3, m_total
- IDC_EDIT5, m_preumoble
-IDC_EDIT4. m_stock
-OnInitDialog
-OnSelchangeList3
-OnOK
-OnButton1
OCLIENT
-DNI
-Nom
-Telefon
-Adreça
-Telefon
-alta_client
-baixa_client
-mostrar_client
-buscar_client
-buscar_client_nom
-modificar_client
VISITA INT.
EDITOR HAB.
VISTES
OMOBLE
-ID
-Nom Moble
-Stock
-Preu
-Tipus(x11)
-alta_moble
-buscar_moble
-buscar_IDmoble
-Generar_Factura
-mostrar_infoMoble
-restar_stock(CString)
-modificar_moble
mostrar_consulta_stock
-preu_taula
OPLANTA
-DNI
-Nom
-DataCreacio
-Preu
-Referencia
-Pagat
-alta_planta
-baixa_planta
-mostrar_llista
-buscar_planta
-baixa_totesplantes
-alta_preu
-marcar_pagat
-buscar_pagat
-buscar_client_pagat
-mostrar_nopagat
-buscar_pagat_dni_nom
-retorna_dni_factura

III.5.4.-Descripció d'interfícies

- Interfície Menú Principal:

El seu nom en el projecte és IClient, ja que al començar la base de dades cada objecte havia de tenir el seu propi menú en una interfície diferent, però més endavant vam decidir que era molt més còmode i eficient accedir a tot arreu des d'una sola interfície principal. A més, la base de dades ha estat en constant reorganització degut a la comunicació amb altres projectes i això ha provocat canvis fins a l'últim moment.

També cal destacar que per qüestions de presentació i de cares a facilitar el correcte enteniment per part de l'usuari hem canviat alguns noms importants, destaquem que tot el referent a habitacions en realitat al projecte amb Visual C++, es deia Planta (Objecte Planta). Ara a cada interfície anirem donant a cada descripció el seu nom autèntic.

Camps:
IDC_MENU_PRINCIPAL: és el menú normal que dona el Visual C++, els seus botons amb funcions són:
ID_CLIENTS_ALTACLIENT
ID_CLIENTS_BAIXACLIENT
ID_CLIENTS_LLISTARCLIENTS
ID_CLIENTS_MODIFICARCLIENT
ID_CONSULTAR_CONSULTARPAGAMENTS
ID_CONSULTAR_CONSULTARSTOCK,
ID_GESTORDHABITACIONS_OBREHABITACI
ID_HABITACIONS_ALTAHABITACI
ID_HABITACIONS_BAIXAHABITACI
ID_HABITACIONS_LLISTARHABITACIONS
ID_MOBLES_AFEGIRMOBLE
ID_MOBLES_LLISTARMOBLES
ID_SORTIR_SORTIRDELAPLICACI
ID_TEXTURES_AFEGIRTEXTURA
ID_TEXTURES_LLISTARTEXTURES
ID_AJUDA_AJUDAPRINCIPAL
ID_MOBLES_MODIFICARMOBLE

Serveis:

- OnClientsAltaclient: Ens porta a la interfície Alta Client.
- OnClientsBaixaclient: Ens porta a la interfície Baixa Client.
- OnClientsLlistarclients: Ens porta a la interfície Llistar Clients.
- OnClientsModificarclient: Ens porta a la interfície Modificar Clients.
- OnConsultarConsultarpagament: Ens porta a la interfície Consultar Factures Pendents.
- OnConsultarConsultarstock: Ens porta a la interfície Consultar Stock.
- OnGestordhabitacionsObrehabitaci: Ens porta a la interfície Obre Habitació.
- OnHabitacionsAltahabitaci: Ens porta a la interfície Alta Habitació.
- OnHabitacionsBaixahabitaci: Ens porta a la interfície Baixa Habitació.
- OnHabitacionsLlistarhabitacions: Ens porta a la interfície Llistar Habitacions.
- OnMoblesAfegirmoble: Ens porta a la interfície Alta Moble.
- OnMoblesLlistarmobles: Ens porta a la interfície Llistar Mobles.
- OnSortirSortirdelaplicaci: Sortir del programa.
- OnTexturesAfegirtextura: Ens porta a la interfície Alta Textura.
- OnTexturesLlistartextures: Ens porta a la interfície Llistar Textures.
- OnAjudaAjudaprincipal: Ens porta a l'Ajuda del programa.
- OnMoblesModificarmoble: Ens porta a la interfície Modificar Moble.

- Interfície Alta Client:

El seu nom en el projecte és IAltaClient.
És on donarem la informació del client per tal de donar-lo d'alta al registre.

Camps:
-DDX_Text(pDX, IDC_DNI, m_DNI)
-DDX_Text(pDX, IDC_NOM, m_nom)
-DDX_Text(pDX, IDC_ADRES, m_adres)
-DDX_Text(pDX, IDC_TLF, m_tlf)
-DDX_DateTimeCtrl(pDX, IDC_data, m_data)
-Botó Accepar
-Botó Cancel·lar

Serveis:
-void OnOK: Comprova que l'usuari hagi identificat el client amb el DNI, que no s'hagi deixat algun camp en blanc, o que el DNI introduït no existeixi ja. Si està tot correcte i s'accepta la confirmació dóna d'alta el client al registre.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Baixa Client:

El seu nom en el projecte és Ibaixaclient.
És on identifiquem el DNI del client per donar-lo de baixa al registre.

Camps:
-DDX_Text(pDX, IDC_DNI, m_dni)
-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
-Botó Accepar
-Botó Cancel·lar

Serveis:
-OnButton1: Comprova que l'usuari hagi identificat el client amb el DNI, que no s'hagi deixat algun camp en blanc, o que el DNI introduït no existeixi ja. Si està tot correcte i s'accepta la confirmació dóna d'alta el client al registre.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Modificar Client:

El seu nom en el projecte és Imodifica.
És on identifiquem el DNI del client per tal de mostrar-ne la informació i posteriorment modificar-la al registre.

Camps:
- DDX_Text(pDX, IDC_ADRES, m_adres)
- DDX_Text(pDX, IDC_DNI, m_dni)
- DDX_Text(pDX, IDC_NOM, m_nom)
- DDX_Text(pDX, IDC_TLF, m_tlf)
- ON_BN_CLICKED(IDC_MOSTRAR, OnMostrar)
- ON_BN_CLICKED(IDC_MODIFICAR, OnModificar)
- ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
- Botó Cancel·lar

Serveis:
- OnButton1: Ens porta a la interfície Llistar Clients.
- OnMostrar: Comprova que l'usuari hagi identificat el client amb el DNI, que el DNI introduït existeixi i està tot correcte es mostra la seva informació als Edits per tal que l'usuari en pugui manipular la informació.
- OnModificar: Comprova que l'usuari hagi identificat el client amb el DNI, que el DNI introduït existeixi, que l'usuari hagi mostrat les dades anteriorment i si està tot correcte i s'accepta la confirmació modifica el client al registre.
- OnOK: Es tanca la interfície.

- Interfície Modificar Client:

El seu nom en el projecte és Imodifica.
És on identifiquem el DNI del client per tal de mostrar-ne la informació i posteriorment modificar-la al registre.

Camps:
- DDX_Text(pDX, IDC_ADRES, m_adres)
- DDX_Text(pDX, IDC_DNI, m_dni)
- DDX_Text(pDX, IDC_NOM, m_nom)
- DDX_Text(pDX, IDC_TLF, m_tlf)
- ON_BN_CLICKED(IDC_MOSTRAR, OnMostrar)
- ON_BN_CLICKED(IDC_MODIFICAR, OnModificar)
- ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
- Botó Cancel·lar

Serveis:
- OnButton1: Ens porta a la interfície Llistar Clients.
- OnMostrar: Comprova que l'usuari hagi identificat el client amb el DNI, que el DNI introduït existeixi i està tot correcte es mostra la seva informació als Edits per tal que l'usuari en pugui manipular la informació.
- OnModificar: Comprova que l'usuari hagi identificat el client amb el DNI, que el DNI introduït existeixi, que l'usuari hagi mostrat les dades anteriorment i si està tot correcte i s'accepta la confirmació modifica el client al registre.
- OnOK: Es tanca la interfície.

- Interfície Llistar Clients:

El seu nom en el projecte és DataBindingDlg.
És on veurem en una taula tota la informació de tots els clients.

Camps:
- DDX_Control(pDX, IDC_DATAGRID1, m_datagridctrl)
- Botó Sortir

Serveis:
- OnInitDialog: Abans d'obrir la interfície, el datagrid (Microsoft ActiveX), es connecta a la base de dades i ens carrega la informació de la taula Access "Client" al datagrid.
- OnOK: Es tanca la interfície.

- Interfície Alta Textura:

El seu nom en el projecte és IaltaTextura.
És on donarem la informació de la textura per tal de donar-la d'alta al registre i per guardar-ne l'arxiu a la carpeta del projecte.

Camps:
- DDX_Control(pDX, IDC_cpotes, c_cpotes)
- DDX_Control(pDX, IDC_tpotes, c_tpotes)
- DDX_Control(pDX, IDC_tbase, c_tbase)
- DDX_Control(pDX, IDC_resp, c_resp)
- DDX_Control(pDX, IDC_proptaula, c_proptaula)
- DDX_Control(pDX, IDC_propcadira, c_propcadira)
- DDX_Check(pDX, IDC_taula, m_taula)
- DDX_Text(pDX, IDC_nomtextu, m_nomtextu)
- DDX_Check(pDX, IDC_cadira, m_cadira)
- DDX_Check(pDX, IDC_cpotes, m_cpotes)
- DDX_Check(pDX, IDC_paret, m_paret)
- DDX_Check(pDX, IDC_sostre, m_sostre)
- DDX_Check(pDX, IDC_tbase, m_tbase)
- DDX_Check(pDX, IDC_resp, m_resp)
- DDX_Check(pDX, IDC_terra, m_terra)

- DDX_Check(pDX, IDC_tpotes, m_tpotes)
- DDX_Check(pDX, IDC_armari, m_armari)
- DDX_Check(pDX, IDC_ase, m_ase)
- DDX_Check(pDX, IDC_Finestra, m_finestra)
- DDX_Check(pDX, IDC_Porta, m_porta)
- Botó Accepar
- Botó Cancel·lar

Serveis:

- Oncadira: Fa que al activar o desactivar el checkbox cadira s'activin o es desactivin també els checkbox referents a les seves propietats.
- Ontaula: Fa que al activar o desactivar el checkbox taula s'activin o es desactivin també els checkbox referents a les seves propietats.
- void OnOK: Comprova que l'usuari hagi identificat la textura amb un nom, que no s'hagi deixat de marcar com a mínim un tipus de textura, que si ha marcat el cadira o el taula hagi especificat alguna de les seves propietats i comprova també que el nom de la textura no existeixi. Després d'acceptar la confirmació s'obre el carregador per indicar la ruta on es troba la foto.BMP de la textura i la copia a la nostra carpeta donant també d'alta la textura al registre.
- void B_Cancel·lar: Es tanca la interfície.

- Interfície Llistar Textures:

El seu nom en el projecte és ILlistaTextura.
És on veurem en una taula tota la informació de totes les textures.

Camps:

- DDX_Control(pDX, IDC_DATAGRID1, m_datagridctrl)
- Botó Sortir

Serveis:

- OnInitDialog: Abans d'obrir la interfície, el datagrid (Microsoft ActiveX), es connecta a la base de dades i ens carrega la informació de la taula Access "Textura" al datagrid.
- B_Sortir: Es tanca la interfície.

- Interfície Alta Habitació:

El seu nom en el projecte és Iplantaclient.
És on donem d'alta una habitació, associant-la al client que identifiquem amb el DNI.

Camps:

-DDX_Text(pDX, IDC_DNI, m_DNI)
-DDX_Text(pDX, IDC_NOMPLANTA, m_nom)
-DDX_DateTimeCtrl(pDX, IDC_DATETIMEPICKER1, m_data)
-ON_BN_CLICKED(IDC_NouPlanta, OnNouPlanta)
-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
-Botó Cancel·lar

Serveis:

-OnNouPlanta: Comprova que el client existeixi, que l'habitació no existeixi, i si és així és dóna d'alta al registre d'habitacions associant-la al client.
-OnButton1: Ens porta a l'interfície Llistar Clients.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Baixa Habitació:

El seu nom en el projecte és Ibaixaplanta.
És on donem de baixa una habitació o totes les que pertanyin al client que hem identificat.

Camps:
-DDX_Control(pDX, IDC_LLISTA, m_llista)
-DDX_Text(pDX, IDC_EDIT1, m_DNI)
-DDX_LBString(pDX, IDC_LLISTA, m_elemllista)
-DDX_Text(pDX, IDC_EDIT2, m_nomc)
-ON_BN_CLICKED(IDC_MOSTRAR, OnMostrar)
-ON_LBN_SELCHANGE(IDC_LLISTA, OnSelchangeLlista)
-ON_BN_CLICKED(IDC_BaixaP, OnBaixaP)
-ON_BN_CLICKED(IDC_baixatotes, Onbaixatotes)
-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
-Botó Cancel·lar

Serveis:
-OnMostrar: Comprova que el client, que s'identifica per DNI i/o per nom, que volem elimiar les habitacions existeixi , si és així mostra el llistat d'habitacions que té associades.
-OnSelchangeLlista: Agafa el nom de l'habitació sel·leccionada.
-OnBaixaP: Si s'ha entrat un client vàlid i s'ha sel·leccionat alguna habitació i està pagada, es dóna de baixa del registe.
-Onbaixatotes: Si s'ha entrat un client vàlid i no té cap habitació pendent de pagar, es donen de baixa del registre totes les habitacions associades a aquest client.
-OnButton1: Ens porta a l'interfície Llistar Clients.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Llistar Habitacions:

El seu nom en el projecte és IListarPlanta.
És on veurem en una taula tota la informació de totes les habitacions.

Camps:
-DDX_Control(pDX, IDC_DATAGRID1, m_datagridctrl);
-Botó Sortir

Serveis:
-OnInitDialog: Abans d'obrir la interfície, el DataGrid (Microsoft ActiveX), es connecta a la base de dades i ens carrega la informació de la taula Access "Planta" al DataGrid.
-B_Sortir: Es tanca la interfície.

- Interfície Afegir Moble:

El seu nom en el projecte és IAltaMoble.
És on dónem d'alta un moble.

Camps:
-DDX_Check(pDX, IDC_ALT, m_altres);
-DDX_Check(pDX, IDC_ARM, m_armari);
-DDX_Check(pDX, IDC_CADIRES, m_cadira);
-DDX_Check(pDX, IDC_CU, m_cuina);
-DDX_Check(pDX, IDC_LAV, m_lavabo);
-DDX_Check(pDX, IDC_LLIT, m_llit);
-DDX_Check(pDX, IDC_LLUM, m_llum);
-DDX_Text(pDX, IDC_NOMMOBLE, m_nom);
-DDX_Check(pDX, IDC_SOF, m_sofa);
-DDX_Check(pDX, IDC_TAU, m_taula);
-DDX_Text(pDX, IDC_PREU, m_preu);
-DDX_Text(pDX, IDC_STOCK, m_stock);
-Botó Cancel·lar

Serveis:
-OnOK: Mira que el moble no existeixi, que tingui preu, un stock i que al menys estigui associat a 1 tipus, si és així, obre el carregador on s'ha de sel·leccionar un arxiu .ASE, aquest és copiat a la carpeta Docs\ASE\Nom_Moble.ASE, aquest arxiu és interpretat i traduït per a poder-lo fer servir en l'aplicació i és convertit a txt, que també es guarda a Docs\ASE\Nom_Moble.txt.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Modificar Moble:

El seu nom en el projecte és Imodifmoble.
És on es dóna un nom d'un moble i es pot modificar el preu i el stock.

Camps:
-DDX_Text(pDX, IDC_EDIT1, m_nom)
-DDX_Text(pDX, IDC_EDIT2, m_stock)
-DDX_Text(pDX, IDC_EDIT3, m_preu)
-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
-ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
-Botó Modificar
-Botó Cancel·lar

Serveis:
-OnButton1: Si el moble existeix ens mostra el preu i el stock.
-OnButton2: Ens porta a l'interfície Llistar Mobles.
-OnOK: Si el moble existeix, modifica els camps del registre del moble.
-B_Cancel·lar: Es tanca la interfície.

- Interfície Llistar Mobles:

El seu nom en el projecte és IListarMoble.
És on veurem en una taula tota la informació de tots els mobles.

Camps:
-DDX_Control(pDX, IDC_DATAGRID1, m_datagridctrl)
-Botó Sortir

Serveis:
-OnInitDialog: Abans d'obrir la interfície, el DataGrid (Microsoft ActiveX), es connecta a la base de dades i ens carrega la informació de la taula Access "Moble" al DataGrid.
-B_Sortir: Es tanca la interfície.

- Interfície Obre Habitació:

El seu nom en el projecte és Ieditaplanta.
És on obrim una habitació per tal de poder-la manipular amb les seves opcions.

Camps:
- DDX_Control(pDX, IDC_LLISTA, m_llista)
- DDX_LBString(pDX, IDC_LLISTA, m_elemllista)
- DDX_Text(pDX, IDC_DNI, m_DNI)
- DDX_Text(pDX, IDC_NOM, m_nom)
- ON_BN_CLICKED(IDC_MOSTRAR, OnMostrar)
- ON_BN_CLICKED(IDC_EDITA, OnEdita)
- ON_BN_CLICKED(IDC_LLISTAR, OnLlistar)
- Botó Cancel·lar

Serveis:

- OnSelchangeLlista: quan escollim un element de la llista aquesta funció ens l'agafa i la va associant a una variable string.

-OnMostrar: Comprova que hagi identificat el client per el DNI o per el nom, si hi ha més d'un client amb el mateix nom demana que s'identifiqui per el DNI , mira si el client existeix i si tot està bé carrega la llista amb les habitacions que pertanyen al client.

-OnEdita: Primer comprova que hi hagi alguna habitació seleccionada a la llista, després amb el dni i el nom els ajunta per agafar la referència de l'habitació i ens l'escriu a l'arxiu PLANTA ACTUAL.txt per al posterior ús dels altres projectes. Finalment ens porta a la interfície Opcions Habitació.

-OnLlistar: Ens porta a la interfície Llistar Habitacions.

- B_Cancel·lar: Es tanca la interfície.

- Interfície Opcions Habitació:

El seu nom en el projecte és Iopcioplanta.

És on escollim quina opció volem utilitzar amb l'habitació prèviament oberta.

Camps:

-ON_BN_CLICKED(IDC_Vistes, OnVistes)

-ON_BN_CLICKED(IDC_EDITA, OnEdita)

-ON_BN_CLICKED(IDC_factura, Onfactura)

-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)

- Botó Sortir

Serveis:

- OnVistes: S'executa el projecte Visita Interactiva
- OnEdita: S'executa el projecte Editor d'Habitacions
- Onfactura: Ens porta a la interfície Generar Factura
- OnButton1: S'executa el projecte Vistes
- B_Sortir: Es tanca la interfície.

- Interfície Generar Factura:

El seu nom en el projecte és IPreu.

És on veiem la factura total de l'habitació i també els preus dels mobles de forma detallada. També és on confirmem si tirem endavant la compra o si només ha quedat com una mostra. També podem marcar aquí la factura com a pagada si el client paga al moment.

Camps:

- DDX_Control(pDX, IDC_LIST3, m_llista)
- DDX_Text(pDX, IDC_EDIT1, m_preu)
- DDX_Text(pDX, IDC_EDIT2, m_iva)
- DDX_Text(pDX, IDC_EDIT3, m_total)
- DDX_LBString(pDX, IDC_LIST3, m_elemllista)
- DDX_Text(pDX, IDC_EDIT5, m_preumoble)
- DDX_Text(pDX, IDC_EDIT4, m_stock)
- Botó Acceptar

Serveis:

- OnInitDialog: Obre el TXT de l'habitació i calcula la factura, mirant els mobles i sumant els seus preus. Ens carrega els noms dels mobles a la llista.
- OnSelchangeList3: Agafa el nom de la llista i en mostra la informació del moble als edits corresponents.
- OnOK: Si la planta ja està pagada i senzillament volíem veure la factura al detall, es tanca la interfície. Sinó, pregunta si es vol acceptar la comanda i si es diu que si, es registra el preu a l'habitació i es resta 1 a l'stock de cada moble. Finalment es pregunta si el client paga al moment per marcar l'habitació com a pagada.

- Interfície Consulta Stock:

El seu nom en el projecte és IConsultaStock.
És on veurem els mobles que ténen stock inferiro a 30, 20, 10 i 5 .

Camps:
-DDX_Control(pDX, IDC_LIST2, m_rang)
-DDX_Control(pDX, IDC_LIST1, m_llista)
-ON_LBN_SELCHANGE(IDC_LIST2, OnSelchangeList2)
-Botó Sortir

Serveis:
-OnInitDialog: Al obrir la interfície mostra una llista amb 4 opcions de stock.
-OnSelchangeList2: Al sel·leccionar una de les 4 opcions de stock mostra a una altra llista els mobles que tenen un stock inferiror al sel·leccionat.
-B_Sortir: Es tanca la interfície.

- Interfície Factures Pendents:

El seu nom en el projecte és IConsultaPagat.
És on veurem les factures que falten per pagar i ens mostrarà també les dades del client a qui pertany la factura. També es pot pagar la facutra.

Camps:
-DDX_Control(pDX, IDC_LIST1, m_llista)
-DDX_Text(pDX, IDC_EDIT1, m_nom)
-DDX_Text(pDX, IDC_EDIT2, m_tel)
-DDX_Text(pDX, IDC_EDIT3, m_adres)
-DDX_Text(pDX, IDC_EDIT4, m_factura)
-ON_LBN_SELCHANGE(IDC_LIST1, OnSelchangeList1)
-ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
-Botó Sortir.

Serveis:

-OnInitDialog: Al obrir-se la finestra mostra una llista amb totes les factues que queden pendents de pagar.
-OnSelchangeLlista1: Al sel·leccionar una factura de la llista es mostraran les dades del client que la té pendent.
-OnButton1: Amb una factura sel·leccionada, podem fer el pagament i així la factura constarà com a pagada.
-B_Sortir: Es tanca la interfície.

III.5..-Disseny de la Base de Dades

Mostrem ara el contingut de les taules del *Microsoft Acces.*

Les taules que hi tenim són:
Mobles, Textura, Client i Planta.

Cal remarcar que els tipus dels mobles, les textures i el pagament de plantes eren inicialment camp booleans, però per problemes d'interpretació del Visual C++ amb aquests camps, vam haver de canviar-ho a camps de text indicant o "si" o "no".

Taula Moble:

CAMP	DESCRIPCIÓ	TIPUS	LONGITUD
ID	Identificador dels mobles	Enter	-
Nom Moble	Nom del moble	Text	-
Preu	Preu del moble	Enter	-
Stock	Stock actual del moble	Enter	-
Cadira	Tipus de moble	Text	2 ("si" o "no")
Taula	Tipus de moble	Text	2 ("si" o "no")
Sofà	Tipus de moble	Text	2 ("si" o "no")
Llum	Tipus de moble	Text	2 ("si" o "no")
Llit	Tipus de moble	Text	2 ("si" o "no")
Armari	Tipus de moble	Text	2 ("si" o "no")
Cuina	Tipus de moble	Text	2 ("si" o "no")
Lavabo	Tipus de moble	Text	2 ("si" o "no")
Altres	Tipus de moble	Text	2 ("si" o "no")

Clau -> ID.

Taula Textura:

CAMP	DESCRIPCIÓ	TIPUS	LONGITUD
ID	Identificador de les Textures	Enter	-
Nom Textura	Nom de la Textura	Text	-
Nom arxiu	Nom de l'arxiu que conté la Textura.	Text	NOM+".BMP"
Terra	Tipus de Textura	Text	2 ("si" o "no")
Paret	Tipus de Textura	Text	2 ("si" o "no")
Sostre	Tipus de Textura	Text	2 ("si" o "no")
Porta	Tipus de Textura	Text	2 ("si" o "no")
Finestra	Tipus de Textura	Text	2 ("si" o "no")
Respaldo Cadira	Tipus de Textura	Text	2 ("si" o "no")
Potes Cadira	Tipus de Textura	Text	2 ("si" o "no")
Base Taula	Tipus de Textura	Text	2 ("si" o "no")
Potes Taula	Tipus de Textura	Text	2 ("si" o "no")
Armari	Tipus de Textura	Text	2 ("si" o "no")
ASE	Tipus de Textura	Text	2 ("si" o "no")

Clau ->ID

Taula Client:

CAMP	DESCRIPCIÓ	TIPUS	LONGITUD
DNI	DNI del Client	Text	9
Nom	Nom del Client	Text	-
Telefon	Telefon del Client	Text	-
Adreça	Adreça del Client	Text	-
Data Alta	Data d'Alta del Client	Data	8

Clau -> DNI

Taula Planta:

CAMP	DESCRIPCIÓ	TIPUS	LONGITUD
DNI	DNI del Client	Text	9
Data Creació	Data de la creació de la planta	Data	8
Preu	Preu total de la Planta	Text	-
Referència	Referència de la Planta	Text	-
Pagat	Data d'Alta del Client	Text	2 ("si" o "no")
Nom Planta	Nom de la Planta	Text	-

Clau ->DNI

Clau ->nom planta

Clau -> referència

III.6.-Exemples d'una Funció:

Exemple d'una alta, en aquest cas, l'alta d'un moble.

```
void OMoble::alta_moble(CString nom,CString cadira,CString taula,CString sofa,CString llum,CString llit,CString armari,CString cuina,CString lavabo,CString altres,int preu,int stock)//, CString data)
{
   int ID;
   CString strConnection = _T("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C://Documents and Settings//FROS23//Escritorio//OpenGL//TFC 4_00//BaseDades//BDTFC.mdb");

        if(m_pDb4.Open(strConnection))
        {
                m_pRs4 = CADORecordset(&m_pDb4);
                if(m_pRs4.Open("Mobles", CADORecordset::openTable))
                {
                        m_pRs4.MoveLast();
                        m_pRs4.GetFieldValue("ID",ID);
                        ID++;
                        m_pRs4.AddNew();
                        m_pRs4.SetFieldValue("ID",ID);
                        m_pRs4.SetFieldValue("Nom Moble", nom);
                        m_pRs4.SetFieldValue("Cadira", cadira);
                        m_pRs4.SetFieldValue("Taula", taula);
                        m_pRs4.SetFieldValue("Sofa", sofa);
                        m_pRs4.SetFieldValue("Llum", llum);
                        m_pRs4.SetFieldValue("Llit", llit);
                        m_pRs4.SetFieldValue("Armari", armari);
                        m_pRs4.SetFieldValue("Cuina", cuina);
                        m_pRs4.SetFieldValue("Lavabo", lavabo);
                        m_pRs4.SetFieldValue("Altres", altres);
                        m_pRs4.Update();
                }
        }
        else
        {
                AfxMessageBox(m_pDb4.GetLastErrorString());
                DWORD dwError = m_pDb4.GetLastError();
                return;
        }
        m_pRs4.Close();
        m_pDb4.Close();
};
```

Conclusions

La primera conclusió és que hem assolit l'objectiu principal del projecte, que era aprendre coses noves. A més, no només d'OpenGL.

Hem après a fer anar el Visual C++, que hem d'admetre, que no ens ha agradat molt. Pensem que et complica molt la vida per fer coses tan senzilles com personalitzar un botó, per fer-ho, s'ha d'afegir codi, classes...Això en comparació amb el seu homòleg, que seria el Borland C++ Builder. També n'és més complicada la connexió i les operacions sobre base de dades, però això ens ha beneficiat, ja que hem après l'ADO.

Per altra banda, ens alegrem d'haver-lo escollit i haver-lo fet servir perquè és un programa molt més lliure. I és per això que aquestes coses en són més complicades.
A més, navegant per gran quantitat de fòrums i pàgines, hem descobert moltíssimes aplicacions i usos del Visual C++, algunes, a molt baix nivell, fet que demostra la llibertat que ofereix.

La base de dades ha estat un afegit que en un principi ni hi comptàvem, ha anat creixent poc a poc al final del projecte. L'OpenGL tenia molta més prioritat. Però hem de dir que ha ampliat molt la visió general del projecte, sent ella la que lliga tots els projectes independents i per tant, resultant ser imprescindible.

Hem creat una petita aplicació de dibuix partint totalment de zero, afegint-hi també opcions com les d'afegir una porta o una finestra que ha complicat l'aplicació.
Ens alegrem però d'haver treballat també amb diagonals encara que també hagin complicat l'aplicació. Hem vist la gran diferència entre les línies ortogonals i la resta i ens ha fet entendre perquè molts programes i inclòs jocs d'aquest tipus utilitzen quadrícules per tal de definir l'espai.

Tocar un programa com el 3DStudioMax, ha estat divertit, ja que són programes que no tenen res a veure amb els nostres estudis, i veure'l i utilitzar-lo una mica ha estat entretingut.

Pel que fa a l'OpenGL, hem de comentar unes quantes coses.
Primer de tot, a partir d'ara, veurem la feina que hi ha darrere de qualsevol aplicació gràfica en tres dimensions. No és gens fàcil realitzar-les, ja que hi entren paràmetres matemàtics molt precisos, s'ha de programar a molt baix nivell (a part de les funcions OpenGL), i a més hi ha el tema del disseny o resultat que veiem al final, que per més

que s'hagi treballat amb una aplicació, al final pot quedar gràficament malament (textures, escenes....).

Nosaltres considerem que encara es pot profunditzar molt més i aprendre molt més d'OpenGL, però podem dir que n'hem après moltíssim.
Hem après a moure'ns, i de forma totalment abstracta , per l'espai i les coordenades 3D.
Sabem crear escenes, carregar-hi objectes, i interaccionar amb ells i amb l'escena en general.
Dins la creació d'aquestes escenes i objectes, hem tocat textures, vèrtexs, filtres, matrius, angles, rotacions i un llarg etcètera. Entren moltíssims paràmetres que hem hagut de veure per tal de fer el projecte.
Dubtàvem de si podríem posar finalment mobles fets amb programes de render, per tal de poder-ne agafar de fets, i incloure'ls per afegir-los i guanyar realisme i vam aconseguir passar del format ASE al format del projecte.

Volem explicar també que d'informació a Internet n'hi ha gran quantitat: tutorials, exemples, fòrums, llocs de codi lliure, etc. Però són bastant repetitius, i bastant introductoris. Exemples que treballen amb un o pocs objectes i que a més estan integrats dins el codi. Ens han ajudat molt i són imprescindibles per aprendre, però un cop assolit lo més bàsic, ens vam haver d'espavilar molt. A més la dificultat augmenta quan amb l'OpenGL, s'executa el programa sense errors detectats, però igualment no funciona bé. Es veu tot blanc o superposat, les coordenades estan malament i surt tot mogut o senzillament no surt, i hi entra també el rendiment de l'ordinador, que no saps fins on aguantarà.

Això també ens ha servit per demostrar-nos de que som capaços d'investigar, trobar i provar diferents recursos. Sempre teníem l'ajuda d'en Jordi Surinyac, que ens ha donat alternatives i orientacions en moments del projecte importants, però dels petits problemes sempre ens n'hem acabat sortint, més o menys a la nostra manera.

Finalment, creiem que la part gràfica ha acabat sent bastant creïble, ja que era una de les coses que desconeixíem com quedaria al principi i que no hem sabut fins pràcticament al final del projecte. Ens alegrem d'haver allargat el període de realització del projecte inicial pel que fa a aquest tema.

Treball futur

Un cop vam arribar al final del projecte i amb tot el que havíem après, se'ns acudien gran quantitat de millores.
Per exemple, afegint portes i finestres, podíem també haver fet radiadors o endolls, etc.
Coses a nivell més general com fer la detecció de col·lisions o treballar més amb les llums i coses més específiques dins el projecte com que el terra seguís la paret, o que les portes i finestres s'integressin ven bé dins la paret. I inclòs arribar a ajuntar habitacions.

El Projecte Vista_Prèvia ens hagués agradat també que mostres els mobles.

La base de dades es pot millorar en molts aspectes i es pot ampliar molt més de com està ara. Un exemple clar, és que no calculi de forma real, els preus dels mobles parametritzables.

Al Projecte Dibuixar també es podria millorar, sobretot també de presentació, amb una bona barra d'eines i afegint-hi més opcions.

Del Vistes poder-ne fer una impressió de les subfinestres una per una o de les quatre al mateix full.

Del Mobles es poden crear "a mà" en un fitxer.txt més mobles parametritzables i ens hagués agradat també, afegir-ne més dels ASE.

A vegades, el programa dona algun error o es tanca directament, suposem que una possibilitat és el tema del rendiment, però també sospitem que en som una mica culpables degut a errors que pensem que fem amb l'optimització. Amb això si que ens hagués agradat tenir més temps per poder-hi profunditzar.

En general pensem que els afegits mai s'acabarien.

Agraïments

Agraïm en primer lloc, a les nostres famílies per haver tingut la paciència de donar-nos el temps necessari per a la realització del projecte.

A en Jordi Surinyac per les seves orientacions, propostes d'alternatives i ajudes. També per haver-nos obligat a fer afegits i modificacions sense els quals el projecte no seria el que ha arribat a ser.

A en Joan Planes per el seu disseny d'interfícies, encara que al final no el poguéssim utilitzar.

A l'Irene Guerola per la traducció a l'anglès del resum.

I finalment, a en Marcel·lí Murillo, en Carles Fajardo i en Josep Maria Benedicto, companys de pis de Girona i Reus, per haver-nos aguantat durant les incontables hores de feina.

Aprofitem també per agraïr als nostres amics i companys de la carrera per haver-hi sigut durant aquests anys a Vic.

Bibliografia

WRIGHT, Richard S.; LIPCHAK, Benjamin . OPENGL SuperBible
Anaya Multimedia-Paperback , 1104 pàg.,3a. Edició. 30 de Juny de 2004.

WOO, M.; Neider, J.; Davis, T.; Shreiner, D.. Open GL Programming Guideversion (Red Book). Versió 1.2, Addison Wesley. 3a. Edició. 1999.

BJARNE, Stroustrup. El lenguaje de programación C++ . Argentina [etc.] Addison-Wesley Iberoamericana cop. . 1993

GARCÍA de Jalón, Javier; RODRÍGUEZ, J. Ignacio; SARRIEGUI, J.M.. Aprenda C++ como si estuviera en primero, Universidad de Navarra, abril 1998.

BUSTAMANTE, Paul; AGUINAGA, Iker; AYBAR, Miguel. Aprenda C++ Básico. Universidad de Navarra, Febrero 2004.

BUSTAMANTE, Paul; AGUINAGA, Iker; AYBAR, Miguel. Aprenda C++ Avanzado. Universidad de Navarra, Febrero 2004.

SÁNCHEZ, J.; AIRES, Felipe. Manipulación del Espacio: Transformaciones y Proyecciones. Universidad de Salamanca, 2000.

GARCÍA, Jorge. Curso de Introducción a OpenGL (v 1.0). Bardock- eghost. 2003.

LINARES, Jordi. Gràfics per computador. Escola Politècnica Superior d'Alcoi. 2004.

JASPE, Alberto; DORADO, Julián. Una Aproximación a OpenGL.

Totes les adreces web que llistem a continuació les hem estat visitant durant gran part del període de desenvolupament del projecte (1/1/2006-1/1/2007).

Visual C++:

http://www.monografias.com/trabajos5/visualcurso/visualcurso.shtml m/

http://www.mygnet.com/articulos/c++/45/

http://www.conclase.net/c/index.php

http://articulos.conclase.net/jm/prog/cpp/cstring.html

http://www.codeguru.com/

http://www.modelo.edu.mx/univ/virtech/prograc/win03.htm

http://www.codeproject.com/

http://www.dcp.com.ar/

http://www.programacionfacil.com/

http://msdn2.microsoft.com/es-es/library/9706cfs5(VS.80).aspx

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

OpenGL:

http://www.opengl.org/

http://nehe.gamedev.net/

http://usuarios.lycos.es/andromeda_studios/paginas/principal.htm

http://www.rush3d.com/reference/opengl-redbook-1.1/

http://pgrafica.webideas4all.com/Contenido.html

http://www.edenwaith.com/products/pige/links.php (ja no està disponible)

http://www.linuxfocus.org/Castellano/March1998/article29.html#section1

http://worldspace.berlios.de/fase1/

http://serdis.dis.ulpgc.es/~itis-iga/PracOGL2002/

http://ascii.eii.us.es/docs/2003-04/videojuegos/docs/6%20-%20Programacion%203D.pdf

http://usuarios.lycos.es/ealonsop/glut.html

http://www.geocities.com/valcoey/tutorial_colision/tutorial_colision.html

http://foros.solocodigo.com/index.php?showforum=21

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

http://www.dsic.upv.es/~fjabad/gpcfi/glutTemplate.html

http://compsoc.dur.ac.uk/kaya/libdoc.php?page=Glut

http://www.lighthouse3d.com/opengl/glut/index.php?10

Textures i Mobles .ASE:

http://www.3dgazpacho.com/3obj.html

http://www.kit3dmodels.com

http://www.coloredhome.com

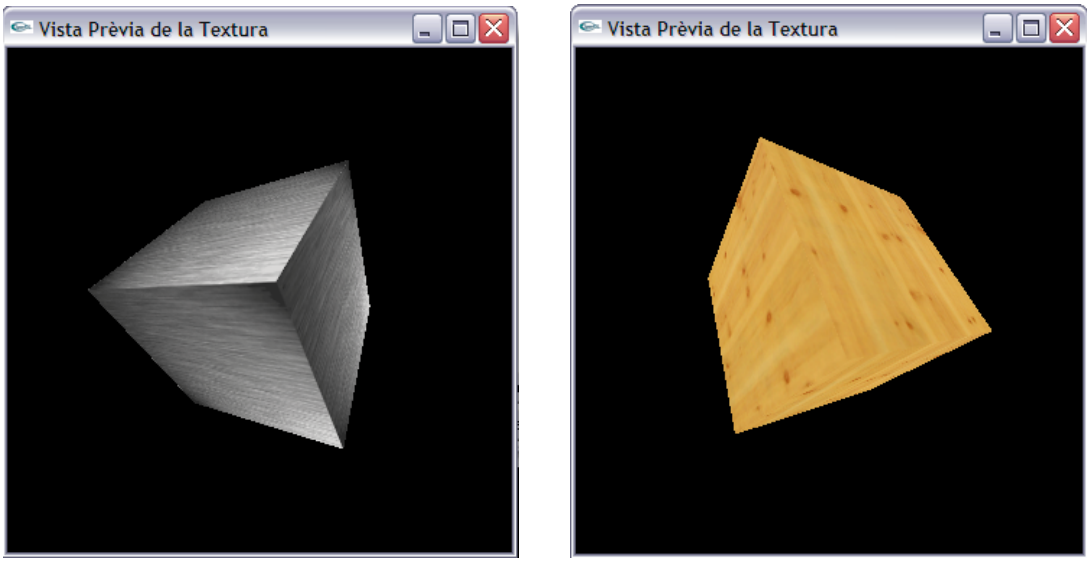
http://www.galiciacad.com

http://textures.forrest.cz

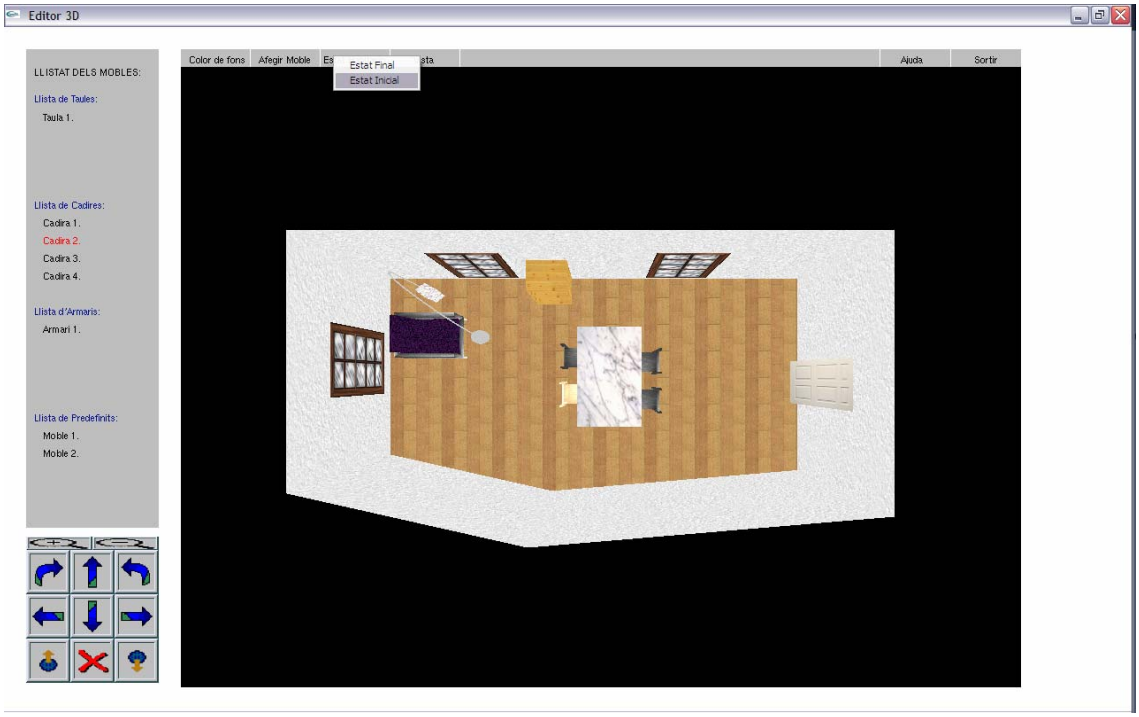
Segurament ens en deixem moltes de pàgines web i tutorials consultats, però hem perdut la referència i per tant no els posem.

Fotografies

Imatge del projete Vistes



Imatges del projecte Vista Prèvia



Imatge de l'Editor d'Habitacions amb una cadira seleccionada

Imatge del projcete Visita Interactiva en Pantalla Completa